Low Latency Rendering with Dataflow Architectures

Sebastian Friston

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Engineering

of

University College London.

Department of Computer Science

University College London

March 13, 2017

I, Sebastian Friston, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the work.

[illegible]

Abstract

The research presented in this thesis concerns latency in Virtual Reality (VR) and synthetic environments. Latency is the end-to-end delay experienced by the user of an interactive computer system, between their physical actions and the perceived response to these actions. Latency is a product of the various processing, transport and buffering delays present in any current computer system. For many computer mediated applications, latency can be distracting, but it is not critical to the utility of the application. Synthetic environments on the other hand attempt to facilitate direct interaction with a digitised world. Direct interaction here implies the formation of a sensorimotor loop between the user and the digitised world - that is, the user makes predictions about how their actions affect the world, and see these predictions realised. By facilitating the formation of this loop, the synthetic environment allows users to directly sense the digitised world, rather than the interface, and induce perceptions, such as that of the digital world existing as a distinct physical place. This has many applications for knowledge transfer and efficient interaction through the use of enhanced communication cues. The complication is that the formation of the sensorimotor loop that underpins this is highly dependent on the fidelity of the virtual stimuli, including latency.

This thesis is concerned specifically with the application of dataflow computing to rendering for virtual reality. Dataflow computing is an alternative computing architecture that distributes an algorithm in space rather than time. The main research questions we ask are how can the characteristics of dataflow computing be leveraged to improve the temporal fidelity of the visual stimuli, and what implications does this have on other aspects of the fidelity. Secondarily, we ask what effects latency itself has on user interaction. We test the effects of latency on physical interaction at levels previously hypothesized but unexplored. We also test for a previously unconsidered effect of latency on higher level cognitive functions.

To do this, we create prototype image generators for interactive systems and virtual reality, using dataflow computing platforms. We integrate these into real interactive systems to gain practical experience of how the real perceptible benefits of alternative rendering approaches, but also what implications are when they are subject to the constraints of real systems. We quantify the differences of our systems compared with traditional systems using latency and objective image fidelity measures. We use our novel systems to perform user studies into the effects of latency. Our low-latency apparatuses allow experimentation at latencies below those previously tested in comparable studies.

The apparatuses are designed to minimise what is currently the largest delay in traditional rendering pipelines and we find that the approach is successful in this respect. Our 3D low latency apparatus achieves lower latencies and higher fidelities than traditional systems. The conditions under which it can do this are highly constrained however. We do not foresee dataflow computing shouldering the bulk of the rendering workload in the future but rather facilitating the augmentation of the traditional pipeline with a very high speed local loop. This may be an image distortion stage or otherwise. Our latency experiments revealed that many predictions about the effects of low latency should be re-evaluated and experimenting in this range requires great care.

Acknowledgements

This research was supported by the EPSRC and Maxeler Technologies Ltd.

The credit for any of this project's success goes to my primary supervisor, Anthony Steed. My development as a researcher has benefited greatly from his skill and patience, and his attitude and approach make our group an enjoyable and rewarding place to work. I am grateful to my second supervisor Simon Julier for all his advice and our thoroughly enjoyable exchanges of all sorts of feasible (and not so feasible) ideas!

My thanks to Maxeler Technologies, especially to my original industrial supervisor Per Karlström. I remember fondly our discussions about video games, and the design of software, hardware and sandwiches. My thanks to my second industrial supervisor Simon Tilbury who always found time to look at the strangest problems (usually of my making). Ashley Nicholls was always fun to be around, and I am grateful to Georgi Gaydadjiev, for looking after the project with enthusiasm.

I want to thank the people I was lucky enough to share our workplace with for the last four years and who have made this experience what it is. Our immediate group (in order of appearance!), Ye Pan, David Swapp, Andrew MacQuarrie, David Walton, Maria Murcia Lopez and Jacob Thorn, as well as Jozef Dobos and Will Steptoe, who have moved on. Also our extended VEIV group, Clement Godard, James Hennessy, Lucy Zarina Campbell and Kelvin Wong. I appreciate Dave Twistleton's continuous assistance, and Graeme McPhillips for the same, as well as our many chats about electronics and otherwise.

Finally to my father, without whom I would not have pursued this, and to my mother, without whom I would not have finished it.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Synthetic environments are those experienced through a computer mediated interface. Synthetic environments may be experienced via a desktop user interface, head mounted display (HMD) or some other bespoke configuration. Like other computer systems, synthetic environment systems consist of a computer, a human operator, and an interface. What distinguishes synthetic environments however is that they aim to transport the operator into a new interactive environment by means of multi-modal interfaces [50]. The first synthetic environments were those experienced by teleoperators, perceiving the real world through cameras and manipulating it through remotely controlled actuators. It was not long though until entirely synthetic or virtual worlds took the place of digitised views of the real one. The purpose of such worlds is to enable more efficient communication, interaction and understanding, by utilising the full range of experiential modalities. Synthetic environments do this by inducing perceptions in their users. They can do this because unlike traditional media, the computer mediation is designed to foster the formation of a sensorimotor loop between the digitised world and its user. Essentially, a mental model of the world is formed, with which the user makes predictions and sees them realised. When this happens they sense the digitised world directly, rather than the interface. These senses are combined into a single perception of the digitised world as a distinct physical place.

The formation of the sensorimotor loop is the underlying distinction between synthetic environments and other media. In order to allow this to take place, the behaviour of the interface must match the predictions of the user. This is complicated by the fact that users will have strong preconceptions about the model, brought with them from their experiences with the real world. Further, in almost all cases, this is not coincidental. Synthetic environments might present a digitised view of the real world in teleoperation or teleconferencing systems, or an entirely virtual environment may deliberately mock the real world with the goal of studying or influencing user behaviour.

Many characteristics affect the perceived fidelity of the model at different levels. The absolute value and range of the light and sound waves is important, but equally so is behaviour of the animate and inanimate objects they represent. Every one of these characteristics is worthy of deep study, but the one this thesis is concerned with is latency. The delay between the input to the interface, and the perceived response, is highly important in synthetic environments. This is because, put simply, most synthetic environments create the expectation of a realistic response - and while the real world has no latency, the typical computers driving such environments do.

That latency limits the effectiveness of computer interfaces is not controversial or novel (e.g. [129, 91, 221, 143, 28, 43, 235]). Such interfaces are limited by the available technology of the time however, and many modalities and characteristics besides latency have had inherent limitations. Recently though, certain techniques and technologies are becoming available to compensate for latency. While latency in synthetic environments will not be eliminated any time soon, it is not unrealistic to consider techniques that attempt to make it imperceptible. The remaining questions are how and where should these techniques be best applied to see this realised.

1.1 Research Problem

Latency is known to inhibit the effectiveness of synthetic environments, yet the presence of latency itself does not mean that the synthetic environment immediately becomes unusable. Interaction with any world, real or digital, and the formation of perceptions is not completely understood. It is a combination broadly of presence [211], the binding problem [134], and the sensorimotor loop [211], with all three concepts having overlaps, interdependencies and ambiguities in their functioning. To work around the lack of understanding in this area, synthetic environments are developed on the principle of correlation between the fidelity of the virtual stimuli and effectiveness. With respect to latency then, the goal is to reduce it by making approximations and assumptions in the simulation, but these must be balanced to avoid introducing artefacts that may have more severe implications than the latency removed. The study of latency is how to make simulations faster, but also the implications of doing so.

Accordingly, this thesis is concerned predominantly with fast computer architectures. It is also concerned with fidelity, more than just temporal, and the impacts of this on user behaviour. There are three main research questions which affect how latency is managed in synthetic environments.

The first is how to make simulations faster, or appear faster. This is essentially the same problem that the creators of real-time interactive virtual worlds have faced since their inception. It is about exploring assumptions and approximations that can be used to optimise away computations, since it is these that cost time. It is also about exploring data structures and how best to store and manage the information within them, to further reduce computations in the critical loop. Examples are taking advantage of temporal coherence, or pre-computation, such as is done in latency compensating image warping.

The second question is how to quantify the fidelity of the virtual stimuli. Fidelity can be judged at different levels, for example by low level response time or dynamic range, to high level physical plausibility [207]. Measuring the physical colour and luminance of the light emitted by a display and comparing it to a capture from a real world equivalent is a way to judge one type of fidelity. An equally important measure however is whether the objects those frequencies represent bounce around the environment in a physically plausible manner, and what their trajectories and interactions convey about their properties.

The third question is what the implications are of the deviations in these from the real world standard. Until computer systems have enough processing power to replicate stimuli indistinguishable from the real world, some approximations and trade-offs have to be made. As before, just because a digitised representation is not perfect, it does not mean it is useless. By understanding how the fidelity of each virtual stimuli affects users perceptions of the digitised world, limited resources can be spent on optimising those characteristics which will have the biggest benefit. A prescient example is image warping, which is becoming very popular in commercial Virtual Reality (VR) systems. This technique minimises perceptual latency at a cost of geometric and spatial artefacts in the resulting stimuli. Understanding the effects of both latency and spatial artefacts independently will allow objectively optimal image warping or equivalent algorithms to be developed - and it is the understanding of both, temporal and the resulting spatial, that is necessary to build the best VR systems.

1.2 Research Questions and Scope

The goal of this project was to explore the applications of dataflow architectures to rendering for VR, and so the predominant area explored was how the advantages of dataflow computing could be used to construct better image generators for Virtual Environment (VE) systems. This was done by building prototype image generators and integrating them into VR systems. As stated above, any optimisations for latency will have implications for other aspects of the stimuli. Therefore a secondary theme of the project was into the characterisation of these effects and how latency affects user behaviour.

The primary questions were if the hypothesised performance of a dataflow rendering algorithm could be realised, when it is implemented ‘for real’. The performance is considered in terms of latency and expected feature set. For example, dataflow graphs are theoretically deterministic, but implementation details such as feedback loops and non-deterministic resources such as Dynamic Random Access Memory mean that there are caveats. What tolerances these theoretical requirements have will determine where a dataflow algorithm may be utilised. Of equal interest was whether such a renderer can integrate with a ‘real’ interactive system and continue to function optimally while subject to its constraints. Both questions are important as they determine where dataflow rendering components may be used.

There are a number of previous works which build similar systems to ours, however the use of dedicated hardware has become less common recently, with many systems opting for implementations utilising the programmable pipeline of off-the-shelf GPUs. Our platform is very high performance, so we are not concerned with building consumer equipment or products, though the lessons learned about the implications could guide such efforts. We do not consider a direct comparison with these previous works, because all such efforts are a means to an end - identifying the requirements of the visual stimuli to get the best VE possible. The final systems we have constructed are most suitable for highly constrained research applications. In terms of making quantitative judgements of fidelity, we make methodological contributions in this area, but deliberately limit our experimentation as this is a very expansive area with much to be done, especially in terms of quantifying temporal characteristics. In this area we borrow heavily from methods used by those working in image compression.

The secondary questions were how the dataflow rendering implementations differed qualitatively and quantitatively from comparable systems using other techniques. As seen in Chapter 5, when examined closely, the temporal behaviour of a VE is too complex to describe with a single value for latency. The objective difference in terms of how the stimuli appears to the user is assessed, as only when this is quantified can the impact on user behaviour be properly measured. Finally, user behaviour is examined at various levels. To get the best trade-off of latency and quality an understanding of how delay and spatial artefacts affect perception, however there is little existing methodology to work from. To begin with, we examine the effects of delay on perception in an effort to identify thresholds and elucidate the effects on important physiomotor functions which contribute to higher order behaviours.

The characteristics that contribute to latency are wide and far reaching. Users do not perceive temporal artefacts from one modality but many. For example the tracking system that measures head orientation will have a different latency to that which measures position, and the delay from the hand tracking will be different to that of the physics simulation that drives the consequences of the moving hands. We constrain our experiments into the effects of latency out of necessity, as such comprehensive experiments would take many theses. We consider only the lowest level temporal characteristics, to do with the sensorimotor loop between the users hand and its proxy, and head movement and perception of the space of the digitised world. Interdependent effects such as the difference in latency between head motion and hand motion are also out of scope and we design the experiments to avoid them. Such interactions are interesting and may be highly consequential, but the implications of low levels of latency (0-20 ms) in the context of a single sensorimotor loop are still not yet understood.

1.3 Contributions

The main contribution of this thesis is the design and evaluation of low latency rendering systems for VR. The prototype image generators presented have been integrated into immersive VR systems and proven capable of underpinning real user studies into the effects of latency. While the inflexibility of the designs mean they are unlikely to find broad applications as a solution to latency, they are a good platform with which to test the effects of latency and image distortions from various sources. Such results can inform the modification and augmentation of existing rendering pipelines to enable the optimal experience. The experiments performed with the apparatus had unexpected results, challenging the prevailing predictions and revealing there is much work to be done before the effects of latency at the current state-of-the-art levels is understood.

1.3.1 Methodological contributions

1. Guidelines for characterising the temporal behaviour of the interaction between rendering algorithms & displays (Chapter 5).

2. An experimental prototype for performing image fidelity comparisons between two synthetic environment systems (Chapter 5).

3. Elucidation of a potentially confounding non-linear effect of latency on the physiomotor loop, with implications for studies that intend to continue using pointing and reaching tasks to investigate low latencies at current state-of-the-art levels (Chapter 6).

1.3.2 Substantive contributions

1. Research findings that address the thresholds of latency on the physiomotor loop in pointing and steering tasks (Chapter 6).

2. Research findings that address the influence of latency on participant gait and distance estimation in immersive VR (Chapter 7).

1.3.3 Technical Contributions

1. Design and prototyping of two ultra low latency renderers, one capable of driving an immersive 3D VR system (Chapters 3 & 4).

1.4 Structure

Chapter 2 contextualises the project and purpose of the research. It provides an introduction to the concept of synthetic environments and how they are applied. A thorough survey of the concepts of presence and the sensorimotor loop and how they relate to synthetic environments is provided, as this is truly the distinguishing characteristic of synthetic environments that underpins their utility. Existing work on latency and rendering in virtual reality is reviewed, so that the current contributions can be judged relative to the experiments on the effects of latency, current approaches to mitigate it, and also to historical attempts to develop similar hardware.

Chapter 3 provides the foundation to the discussions of the technical contributions. It reviews the concept of spatial/dataflow computing and our chosen prototyping platform in detail. This section re-evaluates a number of rendering techniques with respect to the capabilities of dataflow computers. It introduces our first prototype, using it to explore the architecture of rendering systems that map well to dataflow platforms, and the implications of the platform and what its capabilities mean for the design of the algorithm.

Chapters 4 & 5 present in detail a real-time ray caster renderer, built based on our conclusions from Chapter 3. Its design and evaluation is the primary technical contribution of the project. Chapter 4 introduces its architecture and explains its operation, and Chapter 5 describes how it was evaluated.

Chapter 6 presents the first experiment to use one of our low latency apparatuses. The experiment explores the effects of latency on the physiomotor system. Participants interacted with a non-immersive system designed to facilitate the formation of a sensorimotor loop between the hand and a sprite. The experiment used pointing and steering tasks to measure the effects of latency at levels lower than ever tested previously and found the effect was non-linear, in contrast to prevailing predictions.

Chapter 7 presents the second experiment, this one conduced in an immersive 3D world. Participants locomoted in two environments while exposed to differing levels of latency. One environment was designed to induce the distance compression phenomena, and the other to inhibit it. Distance compression is a known phenomena in VR but despite being well studied, it does not have an explanation as of yet. Our experiment was designed to elucidate the correlations between gait and distance judgement with and without distance compression. To our surprise the distance judgements in both environments were consistent with the highest accuracy so far observed in a VE, previously demonstrated only under a very precise set of conditions.

Chapter 8 draws conclusions and suggests future works based on our results.

Appendix A lists the publications that have resulted from this project.

Chapter 2

Previous Works

In this chapter we survey previous works on latency & VR, as well as rendering techniques applicable to VR. We review synthetic environments and what distinguishes them as a concept, as well as the effects of latency and their significance. Understanding what the effects of latency are, and how they come about, is important for deriving specifications of any subsequent hardware. We survey rendering in virtual reality to understand the decisions that have led to the current state-of-the-art configurations and where latency arises in the existing pipeline. We review alternative rendering techniques to glean any available knowledge that may be applicable outside the mainstream, as there may be overlap between these techniques and ours. Finally we review existing specialised hardware and what attempts have already been made to mitigate latency.

2.1 Virtual Reality and Synthetic Environments

Even when computer graphics were limited to vector plotting, and the predominant input device was the typewriter keyboard, innovators and researchers were already as aware as today of the benefits of synthetic environments. Serving as a 'looking glass' into the 'mathematical wonderland' inside the computer, such environments would enable more efficient interaction, experimentation into how humans perceive the world, and exploration of spaces unconstrained by the 'ordinary rules of physical reality' [220].

Synthetic Environments are those in which the interaction of a human operator with a world is mediated by a computer system [50]. Synthetic Environments may be teleoperation systems, in which the computer allows the user to sense and transform the real world. They may be virtual environments, where the world exists entirely within the computer system. In all cases, the synthetic environment system consists of a digitised world, a hardware interface into this world, and a human operator to drive it.

The characteristics of synthetic environments are as highly varied as their use cases. Immersive systems may use Cave Automatic Virtual Environment (CAVE) [38] like environments or HMDs which completely obscure the real world. Augmented Reality (AR) or Mixed Reality (MR) systems combine the real and synthetic through novel optics like half-silvered mirrors. What is common to these, and what distinguishes them from other mediums, is that such systems serve to extend the user's sensorimotor loop - they allow direct interaction with the synthetic world [50, 197].

2.1.1 Applications

This 'direct' sensation and actuation with a computer system has seen clear applications in a number of fields.

Telepresence and Teleoperation has been a mainstay of synthetic environments for decades, as there are tangible benefits of more comprehensive and immersive systems. Devices such as exoskeletons and tactile displays systems allow users to 'feel' the contact between an actuator and the environment (e.g. the contact between a spanner and a nut). Such sensations can significantly improve performance in exploration and manipulation tasks as demonstrated by, for example, Kontarinis & Howe [108, 18]. On the visual side, more immersive representation of sensor data, such as synthesised 3D views, can aid in navigation and reduce operator cognitive workload compared to traditional multi-screen displays, as shown by Lin & Kuo [117].

It is not just physical cues from the inanimate world that can be enhanced either. Synthetic environments can enable the communication of interpersonal cues such as eye gaze [164] and body language [17]. These cues are important - they facilitate more natural interaction and define how one is perceived [223]. In addition to such direct cues, interaction can be enhanced with the perception of shared personal and task spaces [30].

Synthetic environments are also used for training purposes. As explained by Valverde [232], knowledge transfer may occur when two activities are similar in substance or procedure. The ability to recreate sensations and responses akin to the real world has seen synthetic environments adopted for this purpose, especially in areas where operational training may be hazardous or expensive. Some of the first synthetic environments were flight simulators, though virtual reality has been proved effective in a number of equally demanding training scenarios, for example surgery [195].

It is not just muscle memory that these re-created sensations and reactions induce however. The interplay of this sensory data, along with the typical accompanying cognitive processes, forms by the user perceptions - experiences - more than just the sensation of the stimuli themselves. For example, the experiential qualities of immersive virtual worlds have been shown to correspond significantly with the spatial models that they represent [58]. This has seen synthetic environments adopted in areas such as computer aided design and architectural visualisation, to enable better communication and reduce risk [243].

This ability to induce perceptions has seen synthetic environments used for both medical research and treatment as well. For example, virtual environments can be used to conduct psychology experiments that would be too expensive or unethical to conduct in the real world [208]. They have been used to treat disorders, such as phobias with exposure therapy [167]. In some cases, it is simply the perception of ‘being elsewhere’ that has a useful effect, for example as an analgesic during procedures [75].

Finally, a very visible use case is currently entertainment. The ability to directly communicate experiences or places is a facility many game developers have always desired but is now possible [79].

2.1.2 Immersion, Presence & Believability

The previous section contains just a sample of the vast literature on applied synthetic environments. A comprehensive review is beyond the scope of this thesis, if it were even practical. What is apparent from those examples though is that the utility of synthetic environments is beyond simply enhanced performance or communication of information. The practitioners above make reference to *perception*, *immersion*, *embodiment*, a feeling of *'being there'*. These terms refer to a concept that is unique to synthetic environments, and has come to be known as presence [188, 83].

Sherman & Craig [198] describe VR as another stage in the natural progression of technologies utilised for communication and the expression of ideas. They consider the value, not in the medium, but in the experience the media depicts. 'Virtual Worlds' exist apart from the virtual reality system (medium) that delivers them, and their definition of VR emphasises this most important quality of the VE, "giving the feeling of being mentally immersed or present in the simulation".

Until recently there was little consensus on how to characterise this mental immersion. Sherman & Craig define it as the "state of being deeply engaged" and "involved" [198]. This is similar Witmer & Singer's definition of presence - "the subjective experience of being in one place or environment" [249]. Witmer & Singer propose that presence is a result of involvement and immersion. Involvement is a psychological state resulting from a user focusing attention on one set of stimuli relating to an activity or event of interest. Immersion is the perception of being "enveloped by, included in, and interacting with an environment", directly, rather than through an interface [249]. This definition of presence is similar to that of "believability" presented by Kim et al. [105]. Believability is a measure of how much a "participant feels the generated experience as from the real-world". Kim et al. state it is a combination of sensory believability and perceptual believability. Sensory believability is the realism in the "sensory channel" (the fidelity of the stimuli), while perceptual believability corresponds to the realism of the behaviour of the world (the virtual characters to responding as the user would expect, for example). Slater et al. [211] are careful not confound *involvement* and *presence*; the latter they define as "the sense of being there signalled by people acting and responding realistically to virtual situations and events". They assert that presence is independent of involvement or emotional engagement, and is a "reaction to immersion". Immersion here is not a psychological state as it is to Witmer & Singer but an objective measure of the technical fidelity of the VE, defined in terms of resolution, frame rate, FOV, number of sensory modalities, etc.

While there may be some disagreement and overlap in the terminology, there are concepts expressed consistently by all parties.

'Presence is a sense of being, distinct from emotional engagement'

When presence is considered, it is almost always referred to in some way as a perception of being somewhere other than the physical locale (i.e. *in* the virtual world). While there is disagreement to the extent presence and involvement are interdependent, all parties agree that presence is an experience of a user, distinct to the experience of being involved with the synthetic world. An emotive book or film may grab the users attention, but never do they feel they are part of the depiction.

In a VE, they need to be convinced that the photons hitting the retina are not from a display showing a table, but from a table itself. Kim et al. and Witmer & Singer in their respective definitions of sensory believability and immersion consider this to be a result in part of the users focus [249, 105]. Slater et al. consider the acceptance of the virtual stimuli as real to be the definition of presence itself. Emotional involvement, rather than being a contributing factor to presence, can be used to assess it: if the user responds in the same way to generated stimulus as they would to real stimulus, a high level of presence has been achieved [211].

'Presence is achieved through creation of the sensorimotor loop'

Jelfs & Whitelock [92] write that presence is "where we are immersed in a very high bandwidth stream of sensory input" and is "engendered by our ability to affect the world through touch, gesture, voice etc". This ability to affect the world, even if it is just turning ones head to change the view, is the key difference between the feeling of being an observer and a participant. Yet, it is not enough to influence the world: the world must respond in a way that matches the users expectations [249].

Anecdotally it is known that the characteristics of the stimuli for different senses, and the interplay between them, do not equally affect presence. Visual realism, for example, has little correlation with presence, yet dynamic shadows and sound increase it significantly [188]. How the generated stimuli interact to create a sense of presence, or lack of it, is considered by Harvey & Sanchez-Vives [72] to be the same problem as how the brain takes any collection of disparate sensory data to form a unified perception - the *binding problem*.

The binding problem is concerned with how discrete neuronal populations combine information to form a single experience, occurring at all levels of perceptual and motor processing [134]. For example, how is a small, brown, spherical object (modalities of size, colour and shape: all processed separately) resolved into the percept of a ball, and how does the recognition of this object influence the understanding of the current environment?

Harvey & Sanchez-Vives consider cognitive binding, which includes this last question, as most important to presence research. They assert that the loss of presence is a 'failure of binding' owing to a discord between sensory inputs, or sensory inputs and prior experience. Further they suggest neurophysiologic mechanisms responsible for binding can inform the understanding of presence. For example, the theory of hierarchical processing - where sensory details are analyzed at different levels of complexity - could explain the high levels of presence experienced despite low levels of realism [188]. The propensity of the mind to fill-in missing information explains the experiences of participants in simulations of social settings hearing voices, despite none being generated by the VE [72, 211].

It is accepted that participants experience different levels of presence. Witmer & Singer consider this directly proportional to how much attention the participant gives the VE [249]. Slater et al. & Kim et al. recognise the degree of presence as the cognitive level to which the substitution of the virtual stimuli is successful (e.g. sensory & perceptual believability) [105]. Slater et al. define it as "extent that the participant in a virtual or mixed reality forms percepts from the sense data and responds to and acts upon these as if they were real" [211].

While semantic discord in generated stimuli results in reduced presence, it is shown that fast, coherent, plausible stimuli elevates it [211]. Plausible here refers to more than fidelity, but realism in the response of the VE. Slater et al. [211] note the propensity of humans find correlations between their internal state and perception of their environment - creating a model in which observed events are meaningful responses to their actions. They term this *correlational presence*, and the more the VE matches their expectations the more presence is enhanced (*binding* is successful). This takes place at many levels. For example, Garau et al. [66] exposed participants to an identical library simulation containing a number of virtual agents, which responded to the user at different levels (from no reaction, to acknowledging their presence, to attempting conversation). They found with the more complex behaviours, participants expressed a higher sense of presence in a number of ways: in their behaviour (avoiding the 'personal space' of the agent), questionnaire responses, and measured physiological response. At a lower level, Meehan et al. studied the effects of latency at different levels (50 & 90 ms) in an environment designed to provoke a stressful response. While questionnaire results on presence and fear were not conclusive, there was significant interaction between heart rate and latency [143]. The closer the experienced stimuli is to that anticipated by the user (spatially and temporally) the more the user feels the world is responding *to them.* Slater et al. define this as the sensorimotor loop: "the continued, predictable correlation between proprioception and sensory data", which is fundamental to an effective VE [211].

'The sensorimotor loop is a direct result of immersion'

Though the definition of presence may be broad, it has been shown that it is not an unpredictable emergent property, but a direct response from the mind of the participant to a set of stimuli. Though the combination and character of the stimuli set required for optimal presence is still being studied, VEs can be designed, objectively, to maximise presence. Presence has become one of the more popular terms to describe the concept. The idea that the user should respond 'as if the VE were real', has seen the term adopted in literature which assess presence using subjective, objective, high and low level cues and measures [259].

As presence is typically accepted to be a result of immersion, it is easy to see why its most popularly used to describe entirely virtual immersive worlds. This does not mean it is not applicable to other synthetic environments however. The very concept of presence was introduced by Minsky as 'telepresence' where it described the feelings of remote operators of robots beginning to sense they were in a different place, and the robotic avatar becoming part of their own body [188].

Synthetic environments are distinct from other interfaces due to the nature of the interaction that they facilitate - i.e. the sensorimotor loop is formed between the user and the digitised world, rather than the user and the interface. Latency has an obvious affect on this due to requirement of an accurate model which is confounded by latency. The effectiveness of synthetic environments for knowledge transfer and entertainment however rely on more than just low level physiomotor interaction. The concept of presence is underpinned by the sensorimotor loop but may be influenced by other factors such as plausibility and attention. How presence is formed through the binding problem is still unclear. It is clear though that while we are focused on a low level characteristic - latency - we should not ignore higher level functions such as presence, as these are critical to useful virtual worlds. We must also be aware of the potential for significant implications of trading off fidelity for response time.

2.2 Latency in Synthetic Environments and Interactive Systems

One characteristic that appears in numerous studies on the effectiveness of synthetic environments is latency. The implications of inducing presence are what distinguishes synthetic environments, and gives them abilities beyond, traditional media. Presence arises with the formation of the sensorimotor loop. To facilitate the formation users must be able to make predictions of how the world will respond, and see those realised. Such predictions are made in terms of space and time, and so a timely response from the VE is one of its most important abilities.

2.2.1 Definition and Sources of Latency

2.2.1.1 Latency and Jitter

In the context of VEs, latency is ‘the time lag between a user’s action in the VE and the system’s response to this action’ [165]. We consider latency as the total delay of the VE as experienced by the user - from the moment the user begins to move to the moment the photons conveying the consequences of it hit the retina. In addition to the absolute delay from one moment to the next, its change over time can be characterised, defined as jitter. Predictions take time to form, and are based on prior knowledge so jitter is as much of a consideration as latency itself.

2.2.1.2 Sources of latency

The source of latency is the propagation delay of the user input, to the output devices of a VE. This signal is delayed in a number of ways including sampling delay, processing delay, synchronisation or buffer delay and transport delay.

- Sampling delay is the time it takes a sensor device to react to user behaviour. Some sensors may poll the real world at discrete intervals, potentially introducing latency between these times depending on when the stimuli was generated. Alternatively there may be a delay between the user beginning to act and the threshold of a reactive sensor to be reached.

- The processing delay is the time it takes each component of the VE system to transform the input into a form useful for the next stage. For example, a motion tracker may filter and average a number of samples, before transmitting them via USB to the host computer; the host computer will then parse the data from the USB port and form it into system messages to be processed by the VE application.

- Synchronisation or buffer delay results from different stages acting asynchronously. The synchronisation delay is the time between one stage completing its transformation and the next stage sampling the results, due to the requirement of buffering when crossing clock domains [146].

- Transport delay is a result of moving the signal around a system using channels with limited bandwidth.

All of these delays are inherent in any computer system. Faster processors and network connections have a part to play in reducing latency, but in many cases latency is studied so that it may be modelled and thus predicted and compensated for. Non-determinism in any stage will also contribute to jitter.

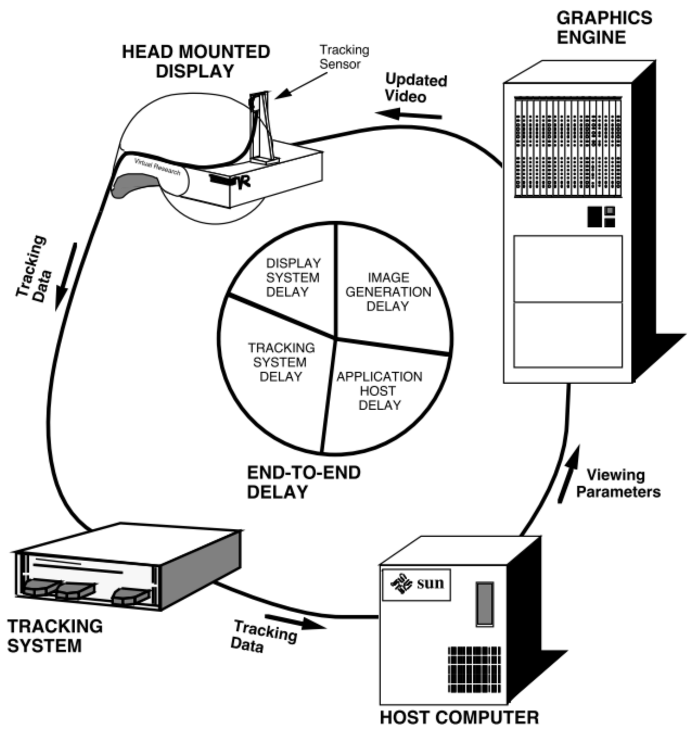


Figure 2.1: End-to-end delays in head mounted systems [147]

Such stages in a typical VE configuration are illustrated by Mine in Figure 2.1. This diagram only shows one such loop however, whereas in practice a VE may have many. For example, the transform of the users head in the world may be a combination of multiple tracking systems, which operate at different rates with different latencies. Further, many components do not operate in direct response to user action, but have a cadence of their own. Examples would be the sampling and polling of a tracker, or the scan-out of a typical display. Where and when a user action takes place relative to this cycle will impact the perceived latency [64]. This is distinct from jitter, which is a characteristic of the system itself.

2.2.2 The Effects of Latency

2.2.2.1 Physical Interaction & Performance

Many studies have been conducted into the effects of latency on physical interaction. It is not hard to see why, given that one of the benefits of synthetic environment is for improving performance and basic interaction tasks usually have unambiguous metrics by which performance can be judged. For example Anvari et al. [9] investigated the impact of latency on telepresence surgery. The number of latencies considered was small, and there were interactions between individual participants (surgeons) and type of task performed, so a model relating latency and performance was not defined. Still, it was demonstrated that latency significantly increased both error rates and task completion time. Jay et al. [91] studied the effects of latency in a collaborative haptic environment, on both performance (with movement times and error rates) and perceived difficulty (with questionnaires). When users were asked to jointly touch and move a target across the screen using a 3 DOF (degree of freedom) haptic interface, they found performance would degrade sharply with latency (starting at 25-50 ms) then level out (after 100 ms for error rate). This phenomenon was explained on examining the questionnaires: they showed perceived difficulty was proportional to latency from 50 ms onwards, which gave rise to Jay et al.'s 'Impact-Perceive-Adapt' model. The theory being that latency affects user performance, sooner than they can detect, and only once latency begins to cause the breakdown of the perception of immediate causality does the user begin to use error-limiting strategies such as moving more slowly.

The same results were found in systems with only visual feedback. Teather et al. [221] studied latency in both 2D pointing tasks and 3D movement tasks, with two input devices (a mouse and a 3D optical tracker). Latency was found to consistently affect the throughput of the pointing tasks, regardless of the input device used. These results were not repeated for the 3D movement task however. It was theorised jitter could have confounded these results.

Other authors attempt to model the effects of latency such that they can predict performance. How the various faculties used to perceive and interact with the world

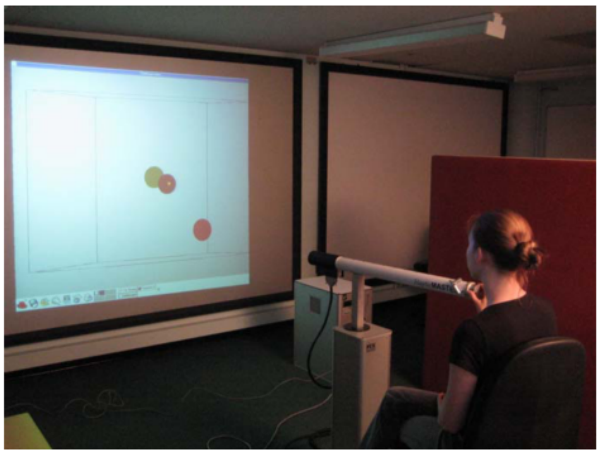


Figure 2.2: Test setup of Jay et al. showing the FCS HapticMaster feedback arm used [91]

operate is not well understood, but some aspects of these can be reduced to the point where they can be accurately modelled and experimented on. The model of how a person carries out goal directed movements such as reaching or pointing has been extended considerably since it was devised by Woodworth in 1899 [251], but the basic principles are that movement can be split into two stages, the ballistic stage and the online stage. In the ballistic stage the brain will launch the muscles in the general direction of a target using information in memory. As the limb approaches the target, the online stage takes over, and the visual system is used more to feedback information to allow the motor system to make fine adjustments and bring the limb directly onto the target [253]. There are a number of interaction models that quantify the relationship between the temporal and spatial characteristics of a person carrying out an interactive task, such as Liu & Liere's model of 3D object pursuit [122], or Accot & Zhai's steering law [1]. The most often studied model however when considering latency is Fitts' Law [56]. Fitts' Law models the motor response of a given person, using two constants (*a* and *b*). With these, it predicts the time (*MT*) to move to any target of given width & distance as shown in Equation 2.1. The the typical apparatus and configuration to study it is shown in Figure 2.3 [253].

$$MT = a + b \cdot ID \tag{2.1}$$

where

$$ID = \log_2\left(\frac{2 \cdot Distance}{Width}\right) \tag{2.2}$$

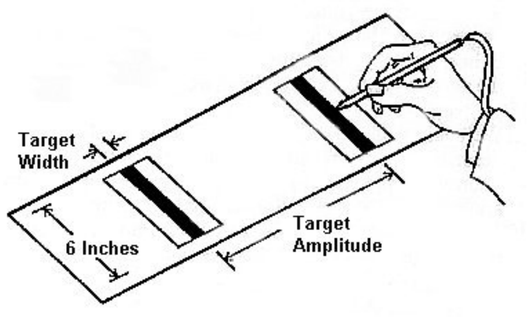


Figure 2.3: Fitts' apparatus for the serial tapping task [128]

Fitts' Law is ubiquitous in studies of HCI (Human Computer Interaction). Its scope & robustness likely play a part in this. It may also be due to it expressing very simply, in its *Index of Difficulty* (ID), where the effect of latency is so keenly felt: the online stage in the above model - the ability to make quick corrections to movement. Conclusive results were achieved by MacKenzie & Ware [129] for example, showing a clear correlation between performance and visual feedback delay in a target acquisition task similar to a Fitts' Law trial, and by Ware & Balakrishnan, who extended the study to include multiple spatial dimensions & lag on different input modalities [240]. In this second study, Ware & Balakrishnan observed the impact on performance of lag in the input device - a 6 DOF hand tracker - and alternatively in the head tracker. They found that performance impact was expectedly large for the hand tracker but negligible for the head tracker. This was not pursued and was mostly attributed to the fact the users made few head movements. Even if they moved more though, the results may not have been that much different, due to the mechanism by which lag is believed to impact performance. MacKenzie & Ware modified Fitts' Law to account for latency, finding it had a multiplicative relationship with ID [129]. So & Chung pursued this and discovered that it was the width component of the ID specifically which latency interacted strongly with (as opposed to the distance, with which there was little interaction) [214].

Watson et al. [242] conducted the first study on jitter and performance, using reaching and tracking tasks. Users were asked to perform these tasks in a VE with varying levels of frame rate[1] and jitter, and their performance in terms of time and error rate was measured. Watson et al. found no significant interaction with jitter and performance: for the high frame rate conditions (20fps) jitter of up to 40% was tolerated without degrading performance, only when the frame rate dropped to 10 FPS did latency begin to interact significantly. The base latency in these conditions was very high however, between 235-285 ms. Further, the jitter was not true jitter, but rather controlled sinusoidal fluctuations with respect to the current frame, making the jitter periodic. Park & Kenyon [166] studied the effects of network jitter on performance in a collaborative virtual environment. In their experiment users completed tasks designed to test dexterity and synchronisation (moving objects through paths), over four different networks (see Table 2.1).

Network	**Latency (ms)**	**Jitter (ms)**
Scramnet-10-msec	10	-
Ethernet	7-18	500
Scramnet-200-msec	200	-
ISDN	150-300	2000

Table 2.1: Network performance characteristics for Park & Kenyon [166]

They found strong interaction between network latency and path difficulty. Though there was no discernible performance difference between the networks for simple tasks, the impact of the choice of network became greater as difficulty increased. This is consistent with the findings reviewed before on latency and the *index of difficulty* of physical tasks. Their results were similar to Watson et al.'s in that no statistically significant interaction between jitter and performance was found. However, they did find the Scramnet-200-msec network resulted in performance closer to the Ethernet network than the ISDN; they asserted more investigation was needed as it not conclusive whether the performance degradation was due to jitter or simply the perceived average increase in latency.

In a study by Gutwin [69], jitter was examined independently of latency to

[1]A proxy for latency, though as described in Section 2.2.1.2 the impact on perceived stimuli is more complex than that of simply increased delay.

determine its effects on prediction. In a collaborative task, one user was asked to move to one of a number of targets as fast as possible, and a second user asked to predict which target the first was aiming for. In a coordination task, users were asked to interact with a shared resource, dragging objects from it onto their own targets. Users performed the tasks with low latency (40 ms) and varied jitter (0-1000 ms), and then with high varied latency (40-1040 ms) and no jitter. Again, jitter had negligible impact compared to latency until high values of jitter (400-600 ms) were reached. Latency had a much stronger interaction with the cooperation task than jitter. Similar results were encountered by Wu & Ouhyoun [162] when they assessed movement prediction algorithms (Linear Extrapolation, Grey-System and Kalman Filter). When provided pre-recorded test data to quantify the various algorithms they found that the Kalman filter had the worst jitter performance, and yet it along with the Grey-System provided the biggest improvements to the performance of the movement tasks participants were asked to complete. The users of these algorithms performed significantly better than those in the control and linear extrapolation groups, with low jitter. The latencies involved in the study were however, like those before it, high: $\sim 120ms$. Anecdotally, both Gutwin and Wu & Ouhyoun observed that when performance of the VE was reduced, users adopted a ‘wait and see’ approach to their actions, consciously waiting for their collaborative partners to finish their movements, and then carrying out their own in isolation.

A study in which jitter was found to be significant was performed by Souayed et al. [216], investigating the use of haptic devices in distributed VEs. Effective haptic feedback must be provided at a much higher rate than visual feedback: ~1 kHz vs. ~30 Hz. In their study of network characteristics on perceived effectiveness of haptic feedback they discovered jitter of 3 ms was unacceptable with a latency of 10 ms.

More investigation is needed into the effects of jitter, especially at lower levels of latency. Though it does appear the visual system has a surprisingly high tolerance of it, jitter has been shown to interact with performance. It is difficult to discern though whether jitter is affecting performance by some mechanism of its own, or whether it is simply on average having the same affect, for the same reasons, as small amounts of latency (the high rates of jitter needed by Gutwin could support this). Further, all the studies referenced here have focused on interactive task performance, which we have seen, relies less on prediction than previously expected. Jitter may have a more considerable affect on presence where the importance of the stimuli matching a cognitive model is higher.

Fitts' Law is just one model of user motion - and readers should note that it is is user motion itself being modelled, rather than the underlying functions. For example, across two studies Samaraweera et al. [187, 186] showed how latency in visual feedback could manipulate gait.

2.2.2.2 Perception & Presence

While the efficiency of physical interaction is important, as we have seen what is truly valuable about synthetic environments is the ability to induce experiences or perceptions of a world that does not exist in the immediate vicinity. A number of authors have examined what the effects of latency are on this. One of the difficulties of this compared to primitive physical interaction tasks is actually measuring presence or perception, which are intangible concepts. Traditionally questionnaires would be used but these must be applied very carefully as they can be subject to interference and mis-interpretation [231, 206]. More recently, physiological cues are being adopted as an objective measure of the users internal state.

Meehan et al. [143] were one of the first groups to consider physiological cues as a measure of presence. They used heart rate to test the effects of latency on presence, under the hypothesis that there would be a stronger effect during exposure to a stressful environment when presence is higher. They found this was the case using a pit-room environment. The results were borderline, however the study was between-subjects with many data points needing to be discarded due to situational constraints. In a meta-study Meehan et al. [142] surveyed four studies which used heart rate and skin conductance, and contrasted these with self-reported measures. Both measures were significantly correlated with factors affecting presence in the expected direction, for multiple exposures and the addition of haptic feedback. Slater et al. [209] investigated the potential of using physiological cues to identify significant changes in the users state. They found for example that participants who had higher social anxiety would have a stronger response to speaking with a virtual character, as signified by heart rate, and that galvanic skin response may have potential to identify *breaks-in-presence*. Breaks-in-presence are the disruptions of presence due to events that cause the users' hypothesis about the environment to fail [205].

In a very comprehensive survey Youngblut [259] reviewed numerous technical and experiential factors that may affect presence. Significant predictors included spatialised audio, image quality, haptics & locomotion technique. However they also included level of interaction/responsibility, level of immersion of a collaborator and task difficulty. These studies all used self-reported measures.

Other authors have used suggested the use of objective behavioural measures, rather than physiological measures, to estimate the users internal state. For example Phillips et al. [176] and Phillips et al. [172] suggested that gait parameters could be used as a presence measure, as the greater the extent to which the user responds to the world as if it were real, the closer these parameters will match the real world behaviour. Phillips et al. [175] demonstrated a correlation between distance judgements and factors that would be expected to increase presence. Mania et al. [131] used spatial memory tasks based on traditional memory research to evaluate the fidelity of virtual environments. Few studies however beside Meehan et al.'s have used physiological or behavioural cues to investigate latency.

2.2.2.3 Experiential Factors

While presence and performance are important, they are not the only ways latency can interfere with the utility of synthetic environments. For example, latency has been shown to contribute to simulator sickness [28], which can be highly disruptive and expensive to training regimes [43]. Anecdotally, latency has resulted in 'Negative Training' when trainees adopted behaviours to compensate for latency in a flight simulator, which degraded their performance when flying a real aircraft [235].

Authors	**Subject**	**~JND (ms)**	**~PSE (ms)**
Ellis et al. [53]	Scene composition and latency detection	10-20	50
Ellis et al. [52]	Hand movements and latency detection	16.7	-
Adelstein et al. [4]	Latency detection mechanism	5-18	5-50
Mania et al. [130]	Latency detection and scene complexity	9.1	14.3
Allison et al. [6]	Tolerance of latency	-	60-200
Moss et al. [151]	Perceptual thresholds for latency	-	147

Table 2.2: Comparison of relative latency detection thresholds

2.2.2.4 Conscious Detection of Latency

We have seen by multiple measures that latency reduces the effectiveness of VEs. We have even seen that it can do so before it is even perceptible to the user. In this section we review when latency becomes perceptible, and under what conditions.

Given that sensory acceptance of the world by the user is dependent on maintaining a sensorimotor loop, it would stand to reason the latency detection threshold would be dependent on the components that make up the loop, and thus the threshold is tightly coupled to the current environment or task. It is surprising then to find that this is repeatedly demonstrated to not be the case. When the JND (Just Noticeable Difference) between two levels of latency is considered, Ellis et al., found it to be independent of the base latency, and of the type of scene being viewed (single object or background) [52]. Mania et al., found no interaction between the JND, FOV, scene content (number of objects) or photo-realism [130]. Moss et al. confirms this, and further that not even the judgement task employed significantly affects the threshold [151]. In another study, Ellis et al. obtained similar results for the JND in a hand movement task (~16.7 ms), as were obtained for those in a head oscillation task (10-20 ms) [53].

The mechanism by which latency is detected then could be fairly consistent across VEs and tasks, which is promising for latency compensation techniques. A summary of the results obtained across various studies is in Table 2.2.

PSE (Point of Subjective Equality) is the level at which 50% of participants detect that the stimulus (latency) has changed with respect to the reference. JND is the additional amount of stimuli required for this level to increase (by convention) to 75%. In terms of psychophysics, it is defined as the *"change in a stimulus required* *to produce a just noticeable difference in sensation"* [130]. Most experiments on latency discrimination follow a similar set-up. A user will wear an HMD and be asked to rotate their head back and forth by a specific angle (selected to keep the visual stimulus within the FOV, usually 30°). Users rotate their head at various frequencies, calibrated by tones from a metronome. They are then exposed to different levels of latency and asked whether they can differentiate between them. This protocol is consistent with that used by Ellis et al., Adelstein et al., Mania et al. and Allison et al. (though the last also used physical buffers to prevent movement) [53, 4, 130, 6]. Moss et al. used a similar configuration but with an interlinked chair and drum, instead of a VE, to provide a system with no latency [151].

In their studies of scene-motion perception Jerald et al. [96] had participants view a projector through goggles limiting the FOV, in order to emulate a zero latency HMD. The purpose of this experiment was to discover the rate at which they could rotate a scene without the user perceiving it. This is distinct from latency detection (the judgement task here is whether there is any motion at all) but the mechanism under study is likely to be very similar if not the same. An often-mentioned discrepancy is between the PSE results of Allison et al. (200 ms) and Ellis et al. (50 ms). Mania et al. first theorised this could be due to scene composition but found this was not the case. Instead they suggest the difference in experiment protocol (the judgement task and level of training) could be the source [130]. Moss et al. pursued this and obtained results similar to Allison et al.: they found the judgement task not to have a significant interaction, but training the users to detect latency improved their performance considerably. Anecdotally Moss discovered that over time participants would devise ways to detect latency (e.g. by focusing on clear fixed features). The exposure time of Ellis et al.'s participants to the VE was greater than those of Allison et al. [151, 52]. Jerald & Whitton in their study of scene-motion perception found different participants had significantly different perception thresholds [95]. The implication being latency discrimination may more similar to presence than physical interaction, in that user characteristics and the 'practice effect' could play a larger part than the underlying biological wiring - and that the threshold may decrease over time.

The 180-320 ms threshold found by Allison et al. is also an extreme result. They studied the interaction of head movement speed on the perception of stability, and found as the velocity of the head increased, the more small amounts of additional latency were reported as causing instability. At 90 degrees per second the 50% threshold was 60 ms, not dissimilar to the results of Ellis et al. [6]. The differences in experimental set-ups between Ellis et al., Adelstein et al., Mania et al., and Moss el al. and Jerald et al. are important in that all VEs have inherent delay. Therefore when latency discrimination capabilities are judged it is with regards to changes in latency, which make the results more pertinent to jitter than absolute maxima. Ellis et al.'s discovery that latency perception does not follow Weber's law, and equally small changes can be detected regardless of the baseline is especially important to this [52]. Out of the above though only Moss et al. have attempted to discover the threshold at which participants can discern between latent and non-latent visual feedback. Though Mania et al. assert since Weber's law does not hold, the sensitivities reported will be equally great compared to zero latency pedestals [130]. Precisely how the latency detection mechanism operates is still an open question. One theory is that it is the result of a discrepancy between the expected view of the world and the experienced view of the world due to image slip. Image slip, or oscillopsia if movement is repetitive, is the phenomena encountered in HMDs whereby the world will appear to move with the user for a latent period, until the VE can update it to its proper perspective (as it would appear if the world were static) at which point it will appear to swim back into position. Image slip, as in the *spatial* discord, rather than the delay between the movement and the motion, was considered by Ellis et al. to be how users detected latency as it accounted for the observed immunity of the mechanism to Weber's law. Adelstein et al. explored this theory by testing latency sensitivity of groups who had predictable head movement, and those that did not. It was found that as participants transitioned from predictable movement to unpredictable movement their sensitivity dropped significantly, but those who started with unpredictable movements showed only a marginal change [5].

Another theory is that it is the swimming - the velocity of the image - that reveals latency. When Allison et al. investigated temporal tolerance they found that latency sensitivity interacted strongly with the speed of movement [6]. Jerald et al. studying scene-motion perception, found that detection rates increased as the world moved against the rotation of the head, or while the head was nearing the end of its rotation (and slowed) - that is, when the relative velocity was highest. They assert that while there are dedicated systems to detect velocity, primate brains rely on cognition to derive acceleration. They also assert that as head movement increases, sensitivity to velocity decreases, explaining both their results and Allison et al.'s [94, 6]. Adelstein et al. explored whether observing displacement error or velocity error resulted in better latency detection. They observed that at the bounds of the head rotation, the relative velocity of the scene was greatest as the head stopped moving but the VE was still correcting, while at the apex of the head movement, the displacement error was high but the relative velocity was low, as the speeds were now well matched. Subsequently, they blanked from view one stage or another, from one of two groups, and found that those observing only the velocity error had much higher sensitivity than those observing only the displacement error, indicating velocity is the main contributor to latency detection [4].

A great deal of work has been undertaken to study the effects of latency on VR. It undoubtedly has an effect, but identifying thresholds has proven difficult and inexact. This is because probing the users' internal state is hard, so it is not always clear whether an effect is due to latency, or a confounding factor, or even if these can be entirely decoupled. Likely for this reason most works focus on low level tasks, which have clear performance metrics. Many works hypothesise the effects of latency on physiomotor behaviour, using observations such as that of latency not following Weber's law. Due to technical limitations though, few have actually explored latency at levels lower than 30-40 ms. Experimental prototypes in area have been refined over many previous works, presenting a obvious avenue to pursue, described in Chapter 6. As seen in Section 2.1 higher level functions such as presence are no less important, though studies into the effects of latency on them have been surprisingly sparse. Experimenters have shown that physiological cues have potential as measures, and hypothesised objective behavioural cues may do so as well. We pursue these in Chapter 7.

2.3 Rendering in Virtual Reality

Synthetic environments aim to exercise the same functionalities as would the real world, so rendering for these is typically concerned with creating a physically plausible approximation of a realistic environment. Rendering realistic 3D environments for VR is one of the most challenging applications due to the extent that the world should be responding to the user. A renderer for a VE should support dynamic lighting & objects, object deformations and a quickly changing viewpoint.

How much visual fidelity affects presence in immersive VR is not entirely clear. Some studies show a strong effect, while others show the relationship is not constant between metrics, with some having no effect at all [210, 264]. Despite this ambiguity, there is a clear trend in VR and related industries towards high quality environments with high pixel densities. This can be seen in design of recent simulators (e.g. [10, 48]). Further, industry leaders predict the eventual adoption of techniques such as ray tracing due to their support of second order effects such as shadows and ambient occlusion, which give high fidelity results [111].

2.3.1 The Rendering Equation

All algorithms which render 3D geometry ultimately approximate the rendering equation. Introduced by Kajiya [102], the rendering equation (2.3) models the radiance (light) from one point in space, visible from another point in space. Or, put another way, the light transported between the first point and the second.

$$I(x,x') = g(x,x') \left[e(x,x') + \int_S \rho(x,x',x'') I(x',x'') dx'' \right] \tag{2.3}$$

The equation states that the intensity I at x from x' can be expressed as the sum of the emittance from x, and the irradiance from all other points visible from x (the visibility is given by the term g). Note how the equation is recursive. If the equation were solved, a model would be created describing the transport of light from every surface point in a 3D scene to every other surface point in the scene. The cost of computing such a dataset is prohibitive, and so rendering algorithms approximate the solution and return a result in a reasonable amount of time. Approximations are made in two ways. First, the accuracy of the light transport model can be tuned to balance quality and computational load. For example, algorithms such as ray tracers will approximate the contribution of all surface points in a scene, by sampling the contribution of a select few. Second, algorithms can pre-process the scene to model the transport of light offline. An example is texture baking, where shadows and colours reflected by surfaces are not determined when those surfaces are drawn on screen, but pre-computed and stored along with the surface. If however the state of the scene changes, for example the shadow casting object changes position, the shadow on the surface will not change leading to a discrepancy which may be detected by the user. Balancing the final image quality, dynamic potential of the scene and rendering time is an ever present theme in computer graphics.

2.3.1.1 Real-time Rendering Continuum

Simulating the physically correct propagation of light in 3D space is prohibitively expensive. However if real-time rendering solutions were limited to only the effects that could be simulated in real-time, the results would be far from the required levels of photorealism. Rendering techniques therefore use pre-computation, to gather and store information about the light distribution in the scene, which then has only to be referenced when the scene is rendered allowing high speeds to be maintained. An example is texture mapping[2] (Figure 2.4). Note the difference in illumination between the front and left sides of the cube - the left side appears in shadow. The shadowing is computed in real-time and the luminance of that side of the cube would change if the cube was rotated. The complex interaction of light with the surface of the bricks though is not simulated, but simply read from the texture in a single operation. The texture, being pre-computed however cannot respond to changes in

[2]Texture mapping is a way to provide local data about points on a surface, beyond that provided by the parameters which define the surface. It can be used for storing light propagation, but much more as well. For example, texture mapping can allow an algorithm to simulate the interaction of light with surface details not stored in the geometric model itself, through the use of normal maps [80].

the scene. As the colour of the bricks has been sampled and stored in the image it would not, for example, provide depth cues as the viewpoint moved above or below the face of the cube.

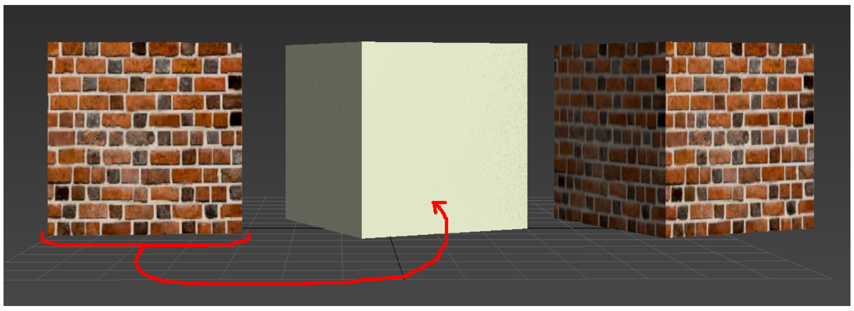


Figure 2.4: Applying a 2D image to each side of a 3D shaded box can create the illusion of geometric details which are not present

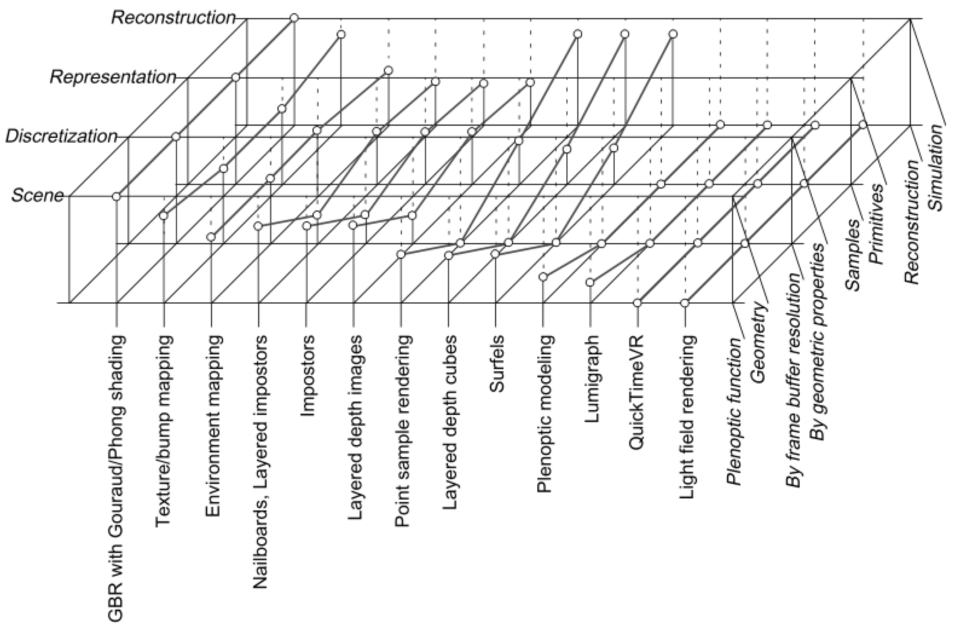


Figure 2.5: All rendering techniques exist along a continuum balancing pre-computation and online simulation to get the best quality images possible within the bounds of the limited hardware of whatever application they are designed for. Image from [265]

All rendering techniques exist on a continuum (Figure 2.5), the contributions of pre-computation vs. simulation in each term of the rendering equation are carefully balanced. The more the lighting in a scene is sampled and not simulated, the less dynamic the scene can be; but the more complex the simulation, the longer the rendering time. This is the rendering continuum and most techniques exist somewhere between the extremes. Zwicker et al. [265] classified rendering techniques with four characteristics: Scene, Discretization, Representation and Reconstruction. Scene refers to how the environment is stored and provided to the algorithm, for example, as functions describing the shape of a continuous surface or a collection of images. Discretization and Representation refer to how the algorithm samples the scene description and how these samples are represented when passed to the renderer. For example, is the scene represented as discrete surfaces or luminance samples. In the final stage, Reconstruction, the algorithm will use these discrete samples of the scene to compute its appearance from a specific point of view. How it does this depends on how the scene is represented. If the algorithm has been provided samples of the appearance of the scene around that viewpoint it may just average them. If it has been provided samples of geometry it must simulate the light transport between them.

As can be seen be seen, with a couple of exceptions, the techniques have a trend from entirely geometry and simulation based to entirely sample and image-based. Over the coming sections, three techniques which span the full continuum will be explained in detail. From the left side of geometry based rasterization - which is the technique of choice for most real-time renderers - to ray tracing, to light field rendering.

2.3.2 Real-time Rendering

Economies of scale have made GPUs the most cost-effective hardware solution for almost every application requiring real-time rendering. Some specialist applications until recently used dedicated hardware, such as boards to perform post rendering warping [235]. Now though, even large simulators are based on Commerical Off-The-Shelf (COTS) parts for their cost and flexibility [10]. Even when new algorithms are presented, they are usually designed for implementation on a traditional GPU [140].

2.3.2.1 Painters Algorithm Rasterization

As a consequence of the ubiquity of GPUs, the most common real-time rendering technique is the one that they have been optimised for - the painters algorithm. The process of rasterization is: taking a continuous geometry representation (polygonal mesh) in 3D space, and processing this into a discrete geometry presentation in 2D space (pixel grid). The ‘2D space’ the 3D geometry is rasterized into, is the frame through which the user sees the scene. Any rendering algorithm which works with a geometry based representation of a scene technically rasterizes it. The distinguishing feature of the painters algorithm is that it projects each 3D primitive into this frame in turn, compositing all elements of a scene together as a painter would. This approach is also the source of the major limitations of this algorithm with regards to latency, and support of second order effects.

An overview of the painters algorithm, or rasterization pipeline, is shown in Figure 2.6. The following section explains the principles behind this method in more detail. An more detailed description of this algorithm and others is available from Hughes et al in [80].

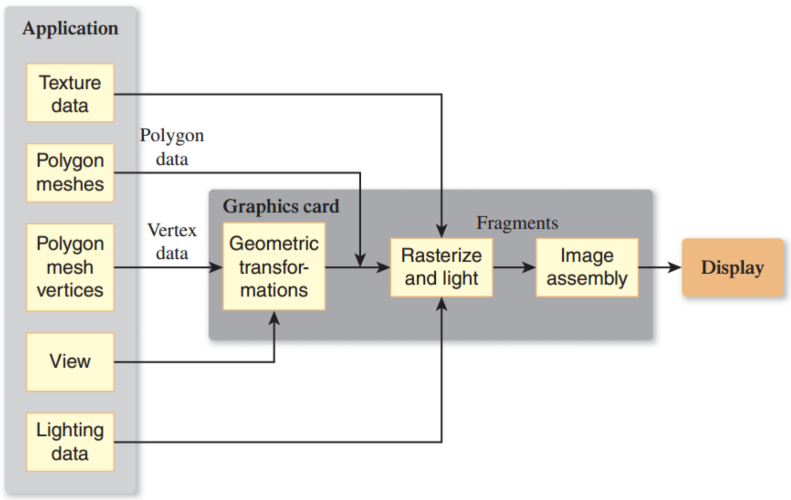


Figure 2.6: Block diagram of the painters algorithm [80]

2.3.2.2 Representing Geometry

The painters algorithm utilises a geometry-based representation of the scene. That is, the surfaces of the scene and their properties are described, along with the lighting conditions. Most commonly the scene will be provided to the rendering algorithm in the form of a triangle mesh. This is illustrated in Figure 2.7, which shows a cube made up of 12 triangles. Each triangle is defined by 3 Cartesian coordinates (known as vertices), all relative to a single point in the scene. The three points lie on a single plane and define the boundaries of the continuous geometry representation.

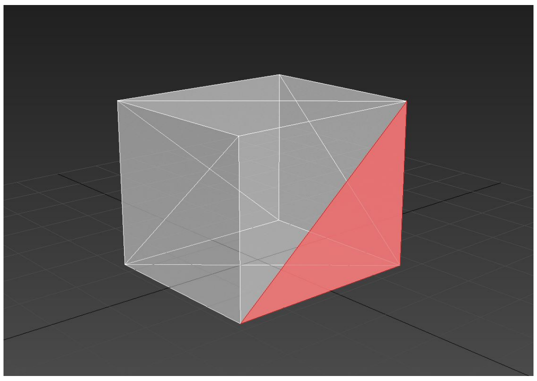


Figure 2.7: A triangle is made up of three points on a plane, sets of connected triangles make up the surface of a 3D shape

While triangle or quad representations are most common they, they are not the only representations. For example, parametric surfaces can be used in place of polygonal meshes. Parametric surfaces are described by a set of control points. A continuous function defines the surface at any given location based on these control points. Such representations may be used in order to achieve a higher quality rendering or a more memory efficient description of the geometry. Algorithms have been designed for real-time subdivision of these surfaces into patches that can map to discrete pixels [34, 37]. In the hardware-accelerated painters algorithm on modern GPUs these representations, no matter what form, are discretized into sets of triangles. This discretization step is termed tessellation. The triangles, whether they came from parametric surfaces or were explicitly modelled, are termed primitives [265, 80].

2.3.2.3 Creating a 2D image of a 3D scene

To render a scene made up of these triangles, a viewpoint or camera must be defined. This viewpoint defines the viewing frustum (image space). This can be thought of as an area of 3D space which contains all visible primitives. It is at this stage that vertices are transformed in world space if required, for example during animation. The transformations for each vertex are provided to the stage, which applies them before projection into image space. This is the ‘Geometric Transformations’ stage in Figure 2.6. The primitives are projected into image space based on the camera properties. Projection is essentially a transformation from one coordinate system (the world space) to another (the viewing frustum). Note that at this point all the primitives are still in 3D space, just not world space. In the next stage, the triangles are projected onto the 2D image plane, and portions of the triangles mapped to discrete pixels. These discrete portions are known as fragments. It is for each fragment that the colour is computed, and these colours are written to the frame buffer to make up the final image [80, 34]. This is illustrated in Figure 2.8.

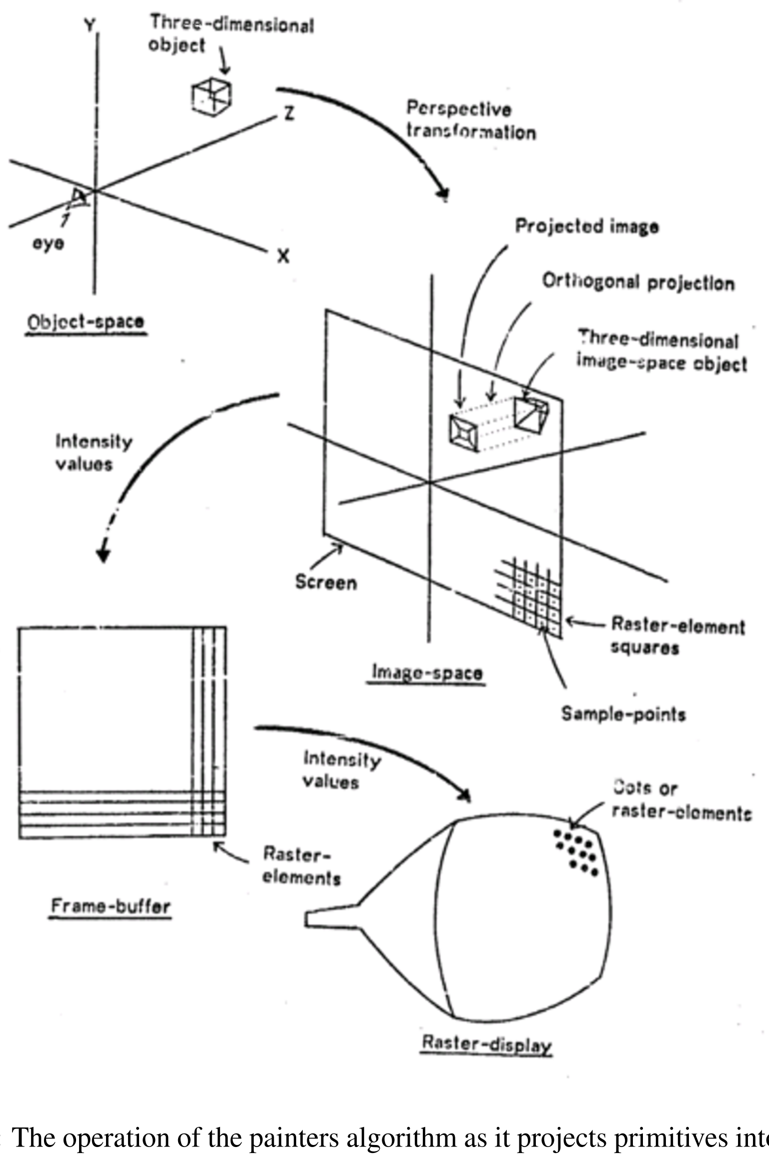


Figure 2.8: The operation of the painters algorithm as it projects primitives into image space [34]

2.3.2.4 Computing the Colour

How the colour is calculated is entirely up to the programmer, but typically a function will approximate the physically correct appearance of a given fragment. This is done by implementing a model of the interaction of the fragment with the lighting conditions of the scene. One such class of models are BRDFs.

A Bi-Directional Reflectance Distribution Function, or BRDF, attempts to model the most salient observations we can make about the interaction of light and matter. We can say, for example, that light is a form of energy and is conserved, so that:

$$light\,incident\,at\,surface = light\,reflected + light\,absorbed + light\,transmitted \tag{2.4}$$

This is illustrated in Figure 2.9.

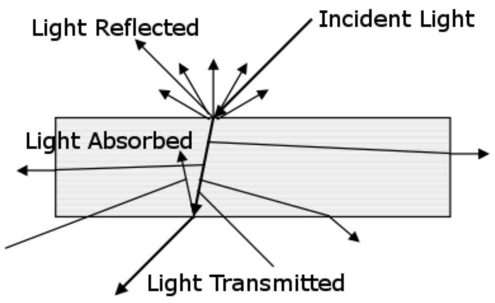


Figure 2.9: Illustration of how light scatters when it meets a surface in the real world [254]

We can also observe the amount of light reflected is not always constant across a surface, but is a function of location. This will cause the radiance to change depending on viewer direction. This can be seen clearly for example in the case of specular highlights (Figure 2.11). The distribution of the reflected light is controlled by the parameters of the model.

A simple BRDF models the radiance from an eye position *e* based on the direction of the light source *l*, surface normal *n* and surface properties (Figure 2.12). Clearly, this is not entirely physically accurate. The function can be made more or less accurate (and complicated & intensive) based on the requirements of the application. For example Sub-Surface Scattering functions can model the absorption

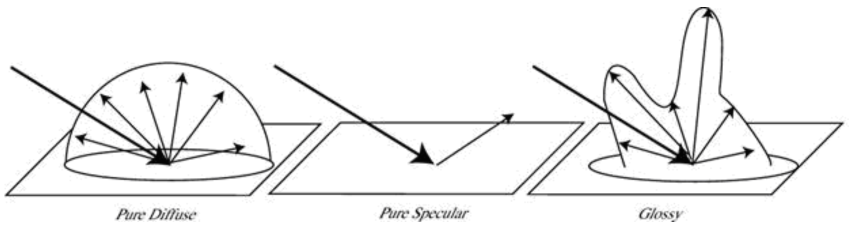


Figure 2.10: Different ways light can scatter depending on the properties of the surface [112]

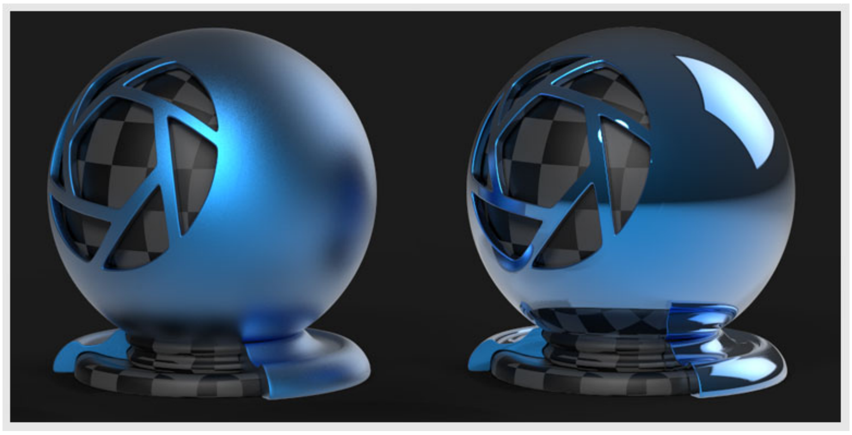


Figure 2.11: Objects showing how specular highlights may appear depending on the roughness of the surface [126]

and re-transmission of light to accurately render materials such as marble and skin. These more advanced functions however usually have a higher computational load that make them unsuitable for real time rendering.

An example of a BRDF is the Phong Lighting Equation given in Equation 2.5 [254]:

$$I_{out} = I_{in}(K_d(max(l \cdot n, 0)) + K_s(r \cdot e)^n) \tag{2.5}$$

I_{out} is the radiance computed by the function. In is the intensity of the light source. l is the direction of the light source to the surface. n is the normal. e is the direction to the viewer and r is the principle reflection direction. These are shown in Figure 2.12. K_d, K_s and n are the parameters that control the distribution of the reflected light and thus the appearance of the material. For example n in this function controls the width of the specular highlight.

In the painters algorithm rasterization pipeline the BRDF or equivalent function

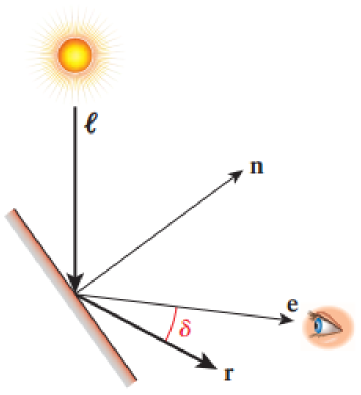


Figure 2.12: Illustration of the parameters a typical BRDF uses to compute illumination for a point [80]

is implemented in the fragment shader. This is the Rasterization and Lighting Block. A shader is a small kernel program executed on the GPU. Each fragment has a position in space, and from this the shader can compute the directions to the viewer position and light source. The properties of the lights and materials are passed to the fragment shader by the programmer, and these are used to solve the function for each fragment. As can be seen, the light sources in the scene are passed only as parameters to the function. They have no other representation or influence on the scene, unless one has been defined explicitly by the programmer. Neither do the fragments interact. High locality and coherence between fragments however allows them to be computed in parallel on a massive scale [254, 80].

2.3.2.5 Image Assembly

Once the BRDF or equivalent has been solved, both the position of the fragment in the final frame, and the colour is known. The fragment cannot yet be written to the frame buffer however. Clearly it is possible for multiple fragments to share the same 2D location in the final frame, as some objects in a non-trivial scene are likely to occlude others. The original painters algorithm does not specify the order in which primitives must be rasterized. The painters algorithm therefore must explicitly handle the case where two fragments overlap.This is done most commonly with the use of a depth buffer or Z-Buffer [80]. When the position of a fragment in image space is calculated, it is calculated in 3D, resulting in a 2D location on the final frame, and a depth value from the camera. The depth value is checked against the current value in the depth buffer, and if it is behind the existing value, that fragment is discarded. The assumption being that an occluding or non-occluding fragment has already been rendered and has written its colour value to the frame buffer. This process makes rendering transparent fragments difficult, with multiple passes needed to ensure that transparent fragments are drawn over opaque ones. More advanced algorithms such as depth peeling have been designed to handle multiple layers of translucent objects, but they add considerably to the rendering time [120].

Transparent objects are the first of many examples of edge cases which must be modelled explicitly outside of the painters algorithm and the BRDF. Shadows, ambient occlusion and caustics are other second order effects which rely on more physically accurate modelling of the transport of light between and within objects in a scene. Once all fragments have been tested in this manner, and had their colours computed and written to the colour buffer, the frame is complete. At this stage post processing effects can be applied. These for example could be image filters, which can easily be applied to the resulting frame in the same way they would be in image editing software. This is one advantage of operating with discrete complete frames. Image Assembly is the final stage in Figure 2.6 and the frame is now ready to be swapped and drawn on the display.

2.3.2.6 Variations on the pipeline

The flexibility of modern GPUs has allowed variations on this pipeline, with data being passed back and forth between stages. For example, deferred rendering involves computing all the fragment positions, and performing visibility tests, before any fragment shader is executed. This allows for increased performance in certain situations as only visible fragments have their colour computed.

There are further opportunities for improving efficiency depending on the application. For example, depending on the BRDF the colour of a flat surface may be constant across it. In this case a fragment program could be executed once for the ’master vertex’, and the result reused for each subsequent fragment of that primitive.

Other techniques include interpolated shading (Gouraud shading). Here the

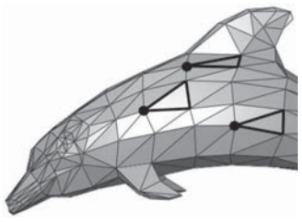


Figure 2.13: Illustration of an object using colour computed from a set of master vertices to shade flat surfaces [80]

BRDF is computed for a set of vertices on a surface, and the colours interpolated between them. This allows curved surfaces to be represented with relatively low resolution meshes [80].

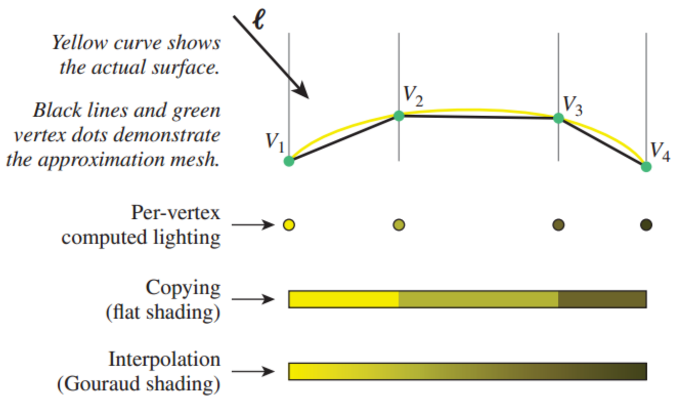

Yellow curve shows the actual surface.
Black lines and green vertex dots demonstrate the approximation mesh.
ℓ
V_1
V_2
V_3
V_4
Per-vertex computed lighting
Copying (flat shading)
Interpolation (Gouraud shading)

Figure 2.14: Illustration of Gouraud shading [80]

All of these variations rely on the feed-forward nature of the painters algorithm. The biggest performance gains come from reducing the number of computations with further approximations. There are no modifications which offer physically correct simulation of second order effects because the surfaces of the scene do not interact. Likewise methods to reduce the rendering time below the interval of a single frame, such as frameless rendering [41], are made difficult by the potential mapping of any primitive to any location in image space.

2.3.3 Modern Graphics Processing Units

Computer graphics algorithms were originally designed to work with very low powered hardware. To begin with all operations were carried out on the CPU. When graphics accelerators first became available operations similar to those described above were implemented in dedicated hardware, with the programmer passing parameters to control each stage. The programmer would pass in, for example, the parameters to the BRDF, and later, vertex transformations. Modern GPUs are essentially stream processors. They consist primarily of collections of compute units (or Single Instruction Multiple Data (SIMD) cores) which are general purpose vector processors. In addition they will have dedicated hardware for accelerating operations such as sorting and texture filtering. The latest GPUs are idiosyncratic devices designed specifically for deep pipelining and execution of rasterization algorithms [80].

It should be noted that the current incarnation of GPUs was not the first, nor the first to become so powerful and flexible that they began to be used as generic coprocessors. Since the 1960's a number of attempts at designing graphics co-processors were made. By requirement these would grow as powerful and fully-featured as the host computer that was driving them. The requirements for low latency access to resources such as (shared) system memory, registers and peripherals would lead to the co-processor moving closer and closer to the CPU, until the overhead of synchronising them made it more efficient to integrate them into a single device. More commonly, the co-processors would adopt features locally, until they became computers in their own right. This led to the need to design subsequent co-processors for the co-processor in order to drive the display, and the cycle repeated. It was referred to, in some frustration, as the "wheel of reincarnation" [152].

2.3.3.1 Parallelism in GPUs

The architectures of modern GPUs include multiple levels of parallelism. They include virtual parallelism, where hardware is timeshared by multiple threads and controlled by a scheduler, and true parallelism, where hardware executes multiple operations simultaneously. GPUs also include a measure of dataflow processing where operations are laid out in space. GPUs present a task-parallel pipeline at a high level to the user, and implement a data parallel pipeline at a low level. GPUs will implement some functions in dedicated hardware. For example fragments can be generated from primitives in image space or texture look ups can be completed outside of the shader programs. The majority of the functionality though is performed by one or more large computational engines. These consist of many SIMD cores, which are vector processors. That is, they perform the same instruction on multiple vector elements simultaneously. The computational engine includes functionality for allocating these cores depending on the load [80, 163].

The architecture of a GeForce 9800 GTX is shown in Figure 2.15[3]. Its computational engine has 16 cores each with 8 data paths. Vector elements will flow through these data paths, with the same instruction being executed on each by the core. Within the data paths long instructions will be pipelined so that the output of one stage of the instruction flows directly into the next. GPU shader programs do support branching, so different sets of cores may process different sets of vectors/vector elements, and the distribution of these sets may change as the shader(s) execute. Managing the allocation of the cores is the responsibility of the Work Queueing and Distribution block (Figure 2.15) [80].

There is considerable data coherency that can be relied in the execution of the painter's algorithm. For example, multiple primitives with the same material properties will be executed in parallel. This is due to the design of graphics APIs, where primitives are split into sets. The programmer will instruct the GPU to draw a set of primitives together, and typically will define the things such as the parameters to the shaders for the set before giving this command. It would take considerable effort on the programmers part to confound this coherency. There are multiple levels of caching in the GPU to take advantage of the coherency, and it further allows some resources such as Texture Units to be shared [163].

As GPUs have advanced, more of the responsibilities of what were dedicated units have been subsumed by the programmable cores. This flexibility along with the independence of each stage of rasterization allows increased efficiency, as multiple pipeline stages can now execute in parallel. For example vertex shaders can execute at the same time as fragment shaders. Subsequent microarchitectures have been

[3]Note that Nvidia's website lists the 9800 GTX as having 128 cores, the 9400 GT has 16. They were both of the microarchitecture family known as Tesla [156].

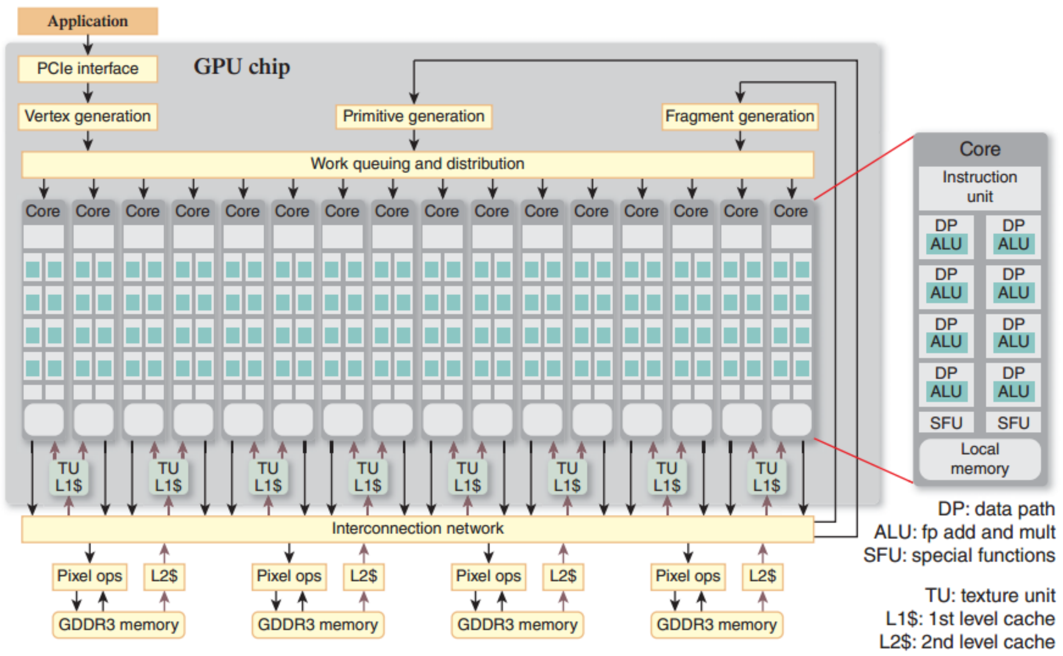


Figure 2.15: Block diagram of the architecture of an NVidia GPU from the family known as Tesla [80]

designed to improve the efficiency with which compute cores are allocated [163]. In Nvidia's Fermi architecture (Figure 2.16), the individual cores were segmented into Streaming Multiprocessors, each with 32 cores and a total of 512 cores between them. Individual threads are combined into a group of 32 threads and executed concurrently by a Streaming Multiprocessor. The Streaming Multiprocessors have their own schedulers which attempt to interleave instructions from multiple thread groups to maximise use of their resources [157, 71, 158].

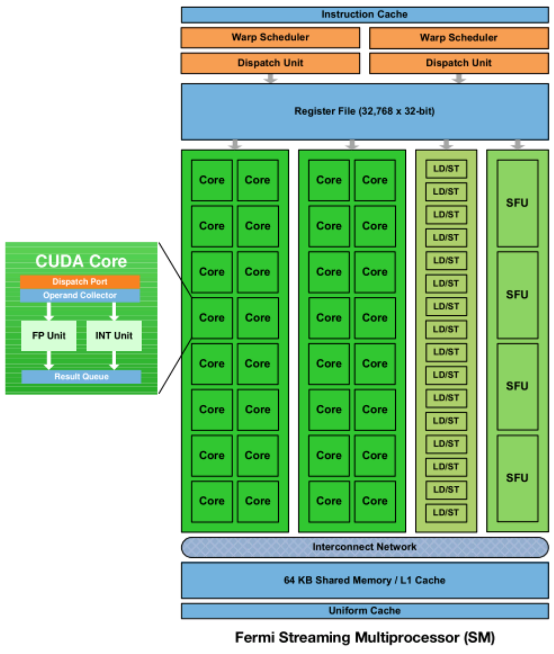


Figure 2.16: Microarchitecture of an NVidia GPU from the family known as Fermi [157]

2.3.3.2 Rasterization and GPUs

As can be seen from Figure 2.15, the rasterization pipeline shown at the beginning of this section maps well to the GPU, with some stages having dedicated circuitry and the interleaving stages being implemented by the main computational engine. Due to the high coherency of the painters algorithm the probability of a set of cores within a streaming multiprocessor executing the same instructions, with the same accesses to external memory is high. Different stages of the algorithm may be executed simultaneously depending on load. For example, if a single triangle took up a large portion of image space, the GPU may execute both fragment shader operations and vertex shader operations simultaneously. The coherency of the painters algorithm is not only important for optimal use of the vector processors, but critical for optimal use of memory. The large DRAM on GPUs has a high latency, in the order of 10s of cycles per access. The closer that data is in cache memory, the longer threads can execute without pre-emption.

There are two main limitations of the painters algorithm. The most important for

VR is that it must assemble the scene representation into a discrete frame. Latency can never be lower then than the interval of a single frame, and “chasing the beam” techniques are not possible. Chasing the beam refers to computing pixel values right before they are required to be transmitted to the display. This is not due to the architectural limitations of GPUs (although they do make implementing alternative algorithms more difficult), but rather the that there is no way to know where in the final frame a primitive will be until it is rasterized. The second limitation is that the scene primitives do not interact. This makes second order effects impossible to approximate in a physically correct manner. Shadows, ambient occlusion and alpha blending must all be explicitly implemented using very different approaches, all of which extend the time it takes to finalise a frame.

Rendering for computer graphics has been refined in a number of respects over time, including the development of highly efficient algorithms and corresponding dedicated hardware. In many respects the characteristics optimised for, such as throughput and dynamism, have overlaps with those required for realistic stimuli for VR. These capabilities have come at the expense of latency however, which is more tolerable in typical systems than it is in VR. The most obvious example of this is the latency introduced during v-sync. V-sync refers to the synchronisation of display scan-out with the updating of the frame-buffer driving it. For example, swapping between back buffers in a double buffered system. This prevents tearing but introduces additional delays before the latest computations become visible to the user. The frame-based nature of the rendering techniques used is the most obvious place to start optimising for latency. As many of the techniques should be re-used as possible though as there are overlaps with the requirements for VR, and ultimately the same problems are being solved for both applications.

2.3.4 Image Warping

A post-rendering technique that has been of interest in computer graphics for some time is image warping. Its applications are varied, including image transitions, stereo view synthesis and temporal up-sampling. It has held particular interest in VR however, due to its potential for latency compensation [132]. Image warping could be seen as a subset of image morphing. Image morphing is the process of metamorphosing between two images using a mapping function that defines the spatial relationship between the points in the images [250]. For latency compensation, the morph is between a rendered image, and an image of the same scene but with a different viewpoint. Essentially, image warping is used to generate a novel view of a scene from a previous render in order to compensate for changes occurring during the rendering process itself. Image warping is currently used this way in commercial VR systems [159].

Image warping is attractive for latency compensation because the mapping functions can be simple and highly parallel. For example, Yanagida et al. [256] used a 2D image shift, suggesting it may be suitable for local compensations and could be implemented on the HMD itself. As illustrated by Wolberg [250], the mapping function itself is typically independent of the scene content. However, it is generated based on image primitives and features. Image warping algorithms therefore have various levels of coupling to scene content and scene representation.

A dominant challenge for image warping is disocclusion. For novel view synthesis or re-projection with non-linear mapping functions, there is the potential that not be enough data will be present in the source image to sample for a given destination point [190]. This can be illustrated by imagining a camera moving around a large item such as a desk or cupboard, revealing more of the environment behind it. The content just come into view will not exist in the source image because this location was not visible before. A simple bijective warp such as Yanagida et al.'s image shift would not require this data, but would instead introduce distortions in perspective cues [256].

To handle these artefacts, some authors use selective-rendering. For example Widmer et al. [244] use depth-peeling to render a set of planar approximations of the scene geometry. These could be used for screen-space ray tracing for fast in-painting, but also other effects such as multi-bounce specular and glossy reflections. The issue with using these approaches for latency compensation is that it further reduces the determinism of the rendering process. As such, authors have attempted synthesis using only existing samples, either by optimistically rendering additional samples, or using additional data about them.

For example, Reinert et al. [182] presented a pipeline for low-power applications such as phone-powered VR. They transmitted a remotely rendered environment as an overscanned hemispherical environment map. This supported straightforward image warping for changes in rotation. However, they also transmitted a lower resolution alternative view of the scene. This was designed to maximise visibility of surfaces occluded in the primary view. This could provide samples in cases of disocclusion during translation warps of the primary view.

Nehab et al. [154] presented a caching system to allow re-use of shading calculations between frames. This system is based on reverse re-projection, in which fragments in a current frame are re-projected into a previous frame to determine if there is an existing sample available. This saves time for all shading calculations, but may be particularly beneficial for techniques such as expensive global illumination approximations.

Yang et al. [257] stored multiple adjacent frames with depth information. In the case of missing pixels, they used an iterative search for the sample with the nearest depth value out of a set of adjacent frames. This search was facilitated using motion flow fields. These define the linear motion of each pixel in the re-projection. This could be used to direct the search through the adjacent frames' depth buffers. Bowles et al. [25] also used motion flow fields to generate an image warp. In Bowles et al.'s implementation, disoccluded areas were in-painted using the surrounding background. How well this works depends on the background itself.

Didyk et al. [46] performed image warping for stereo view interpolation. They used disparity information across the frame to generate the warp. Their implementation represented the source image as a set of textured quads - areas rather than pixels. These were then transformed into the warped image. Doing so automatically handled disocclusion by stretching the quads to cover missing areas of the frame. Didyk et al. [45] also applied the technique to image warping for temporal up-sampling, but using motion flow instead of disparity maps. They used selective blurring to hide potentially salient artefacts, taking advantage of the natural de-blurring behaviour of the human visual system during motion tracking to create the perception of sharp, high quality images. The authors demonstrated the utility of their implementation by showing significantly improved user performance in an moving object classification task.

Image warping has been explored for temporal up-sampling for some time, and has potential for latency compensation. The advantages of image warping are that the warping function itself is usually decoupled from the scene geometry. This makes the technique amenable to parallelisation and even hardware implementation [235]. The potential for disocclusion and perspective cue distortions in simple warping functions however mean many such algorithms are not truly independent of the scene content and must rely on, for example, iterative searching for suitable samples.

The larger the distortion the higher the probability of salient artefacts. For this reason image warping for VR has been applied so far in narrow applications close to the display, where the times compensated for are low ($< 10ms$). An example is Oculus' predictive warping functionality, which predictively warps an image to compensate for both rendering and scan-out delays reducing the perception of latency [16].

2.4 Alternative Rendering Solutions

2.4.1 Ray Tracing

2.4.1.1 Why Ray Tracing and Path Tracing?

Ray Tracing is an algorithm that models the transport of light through a 3D scene. The painters algorithm computes the appearance of each object, and projects this into the camera. Conversely, a ray tracer samples the scene by looking out of the camera, identifying what points are in front of the viewer, and computing their colour based on the interactions between light emitters and other objects. Ray Tracing is a very popular rendering technique in a variety of applications. This is because it is conceptually simple, and provides considerable flexibility in the implementation. Yet while simple, ray tracing is based on modelling a physical phenomena, and if the mathematical models employed by the ray tracer are sufficiently advanced the results will be very high quality. Ray Tracing is the technique of choice for commercial rendering engines such as mental ray[4] and VRay[5], and in-house tools such as Renderman[6]. It is also employed in some real-time graphics solutions as a part of pre-processing stages such as light mapping [229]. Ray Tracing has been suggested as the next stage of computer graphics and some companies are developing real-time ray tracers[7] [111].

In the next section we discuss ray tracing, path tracing and optimisations of these techniques, such as the use of acceleration structures. Ray Tracing principles such as the basic mechanics of casting a ray are used in a number of computer graphics algorithms, especially in various image-based rendering techniques. When we use the term ray tracer however we refer to an implementation that computes the colour by modelling light transport in real-time.

2.4.1.2 Principles of ray tracing & scene sampling

Ray Tracing works by numerically solving the rendering equation. It samples the radiance at discrete points in the scene, by using rays between surfels to compute visibility, and, along with the BRDF of the surfaces, the transport of light between the surfels.

A frame is rendered by defining a viewing plane in 3D space based on the viewpoint. From each pixel on this plane, a ray is cast into the scene and is tested for intersections with the scene geometry. When an intersection is found the colour of that point is computed based on the geometry surface's BRDF and irradiance.

How the irradiance is approximated is one of the biggest design decisions in implementing a ray tracer. There are various ways to approximate the irradiance. For example, in Monte Carlo Ray Tracing the irradiance is computed by sampling the environment. For each intersection of a primary ray, a number of secondary rays are cast to sample the surrounding environment and compute the most statistically

[4]http://www.autodesk.com/products/mental-ray-standalone/overview
[5]http://www.chaosgroup.com/
[6]http://renderman.pixar.com
[7]http://http://www.render.otoy.com

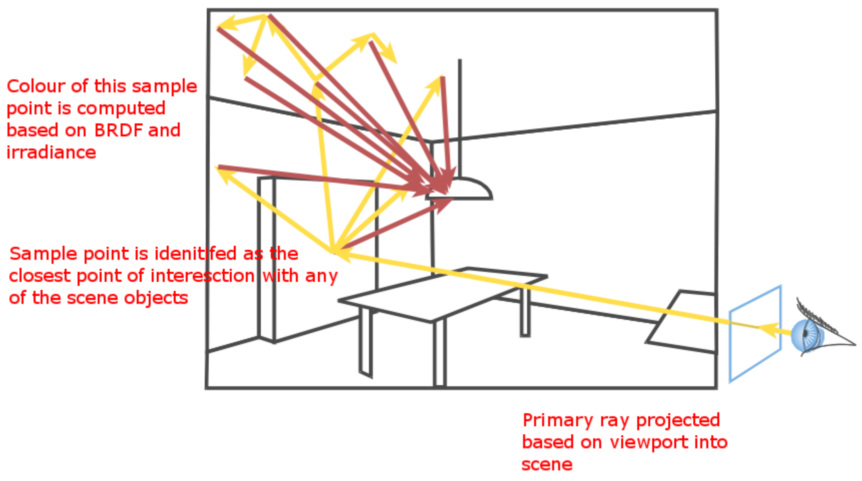

Colour of this sample point is computed based on BRDF and irradiance
Sample point is idenitifed as the closest point of interesction with any of the scene objects
Primary ray projected based on viewport into scene

Figure 2.17: Illustration of Monte Carlo Ray Tracing [40]

probable contribution of light. The rays are selected so as to distribute the sample points uniformly above the sample. That is, the sample points are distributed evenly, but not repeatedly. This is because repeated patterns in sampling can result in aliasing or other artefacts. Figure 2.17 illustrates how the colour of a pixel is computed using this method.

In the simplest implementation, irradiance is not calculated at all. The (assumed un-occluded) ray between a sample point and each light source in the scene is computed and used to solve the BRDF, just like it is in the painters algorithm. It is a misconception that ray tracing includes second order effects 'for free'. Second order effects such as shadows are included 'for free' depending on the implementation. For example, Monte Carlo Ray Tracing involves a large number of samples per point, and the differences in colour between these based on the interactions it models will naturally approximate shadows and global illumination. The more deterministic alternative is to produce these second order effects explicitly. For example, instead of assuming the ray between the sample point and the light is un-occluded, this could be tested by looking for intersections with occluders in the scene along the ray. This will increase the computational load however, as it goes from $scenecomplexity * framesize$ to $scenecomplexity * framesize * numberoflights$. Similar approximations can be made for ambient occlusion. For example by sampling a number of rays about a point. Again it is clear that the time per sample will increase dramatically as this is carried out.

2.4.1.3 Computational requirements of ray tracing

The main operations involved in ray tracing are intersection tests and solving the BRDF. An example of a plane intersection test for a ray is shown in Equation 2.6 [193]:

$$t = \frac{N(A,B,C) \cdot O + D}{N(A,B,C) \cdot D} \tag{2.6}$$

Where the ray is expressed as

$$R = O + t \cdot D$$

O is the origin in 3D space, D is the direction vector and t is the intersection point along D. From Equation 2.6 it can be seen that these tests are solved analytically, not numerically, and so are deterministic. t is computed for every primitive in the scene, and the lowest value of t defines the closest (and therefore valid) intersection point [219]. The BRDF is of arbitrary complexity depending on how accurate the developer wishes it to be.

The computational load is linearly dependent on the number of rays cast, the complexity of the BRDF, and the number of primitives in a scene. The number of rays cast is proportional to the frame size, and quality. Quality in this case refers to the minimisation of artefacts, and the inclusion of second order effects. Both of these are typically solved by casting more rays. The number of rays can increase laterally or depth-wise (that is, multiple parallel samples, or multiple levels of recursion). This decision will alter not just the performance but also the kind of effects that the ray tracer will support. For example, caustics can only be modelled with path tracing (multiple levels of recursion). Path Tracing also shows fewer artefacts than radiosity-based implementations. The two techniques are different enough to be referred to by different names [40, 93]. Ray Tracing is highly parallel with high data locality, but very high memory access requirements. Individual rays do not need access to each other but need repeated access to the representation of the scene. Control flow requirements are high although how much so is implementation dependent. Some implementations, such as packet tracing, create groups of spatially coherent rays to be traced together, while others allow the rays to diverge quickly [228, 238].

2.4.1.4 Optimising for ray tracing

Acceleration Structures

As scenes grow in complexity, acceleration structures are practically prerequisites for ray tracer implementations. Acceleration structures are entirely concerned with reducing the number of intersection tests. Most acceleration structures reduce rendering time by culling primitives from the list of potential intersections, but do not remove the requirement to loop over a great number of primitives or traverse the structure in a non-deterministic manner. Acceleration structures are typically implemented as a tree or grid, and the scene primitives partitioned into these so the intersection tests are performed with subsets of the scene. That is, the acceleration structure is used to cull scene geometry before intersection tests are performed [238]. Since the type of acceleration structure will depend on application requirements, e.g. static vs. dynamic geometry, and we are most interested in deterministic algorithms which map well to our dataflow platform (Chapter 3), we do not discuss these in further detail.

Ray Coherency

Along with acceleration structures, many ray tracing implementations use ray coherency to reduce the total number of operations required to render a scene. This is especially common in ray tracers implemented for SIMD architectures such as GPUs. Ray coherency refers to the fact that many rays are likely to follow the same path in space. Based on this knowledge, ray tracers can aggregate rays into a packet. Operations including intersection tests and memory accesses can be performed for the packet instead of each ray individually [24]. Alternatively, the same operations can be performed for each ray in the packet but on a SIMD architecture. The results of the operations on all rays are available but the execution time remains the same. The high data locality between the rays in the packet also enables optimal use of local caches [239]. Considering a set of rays as a packet also provides the opportunity to use operations that do not apply to rays. For example, a ray tracer could use interval arithmetic to perform intersection tests with an acceleration structure, or even primitives themselves. Interval arithmetic involves performing basic numerical operations, but with intervals (in $\mathbb{R}$) instead of real numbers. The intervals in this case would be the bounds of the ray packet. Alternatively the area of the packet could be described as a primitive, and things like geometric intersection tests used in the place of ray intersection tests [107, 65].

Another optimisation which utilises spatial coherence is beam tracing. This was introduced by Heckbert & Hanrahan in [74]. Instead of casting rays in world space, the beam tracer transforms the world geometry into beam space, then uses the volume of the beam to find intersecting polygons (in the same way they would be projected into image space). The polygons are then ordered by depth to find the first intersections with the beam. As in ray tracing, the beam tracer proceeds recursively from these intersection points. This is analogous to placing a camera at the beam origin and projecting the polygons of the world into this. The viewing frustum being the same as the beam geometry. The authors used a modified painters algorithm to compute the irradiance at specific points in the scene. The irradiance was used with the BRDF of the polygons to compute the final colour of those points to render the scene in a similar manner to a ray tracer. They also suggest inverting their beam tracer to bake illumination, by projecting beams outwards from light sources.

Photon mapping and other effect-specific pre-computation techniques

As explained at the start of the section, the use of ray tracing does not imply support of second order effects. Where shadows and ambient occlusion are required, they can be approximated by casting extra rays. Alternatively, there are acceleration techniques dedicated to these. One such technique is photon mapping. A good overview of photon mapping is available in [93].

Photon maps are structures describing the distribution of light in a 3D scene. They are produced by emitting rays from the light sources in the scene, much like rays are cast from the camera when the scene is rendered. Where these rays intersect geometry, a photon is stored in the structure describing in the irradiance from the light source at that point. A probability distribution along with BRDF of the surfel the ray intersected with, is used to produce a set of rays which are emitted from this point, and subsequently model the irradiance to other parts of the scene. This process is repeated recursively for as long as the photon mapping algorithm dictates. Photons are stored in structures such as KD-Trees.

The photon map stores the irradiance at all points throughout the scene. A traditional ray tracing algorithm then, when resolving the rendering equation could look up from the photon map the irradiance term. (Lookup, or resample based on surrounding photons.) Further, photon maps can be used to guide the distribution of further rays. A photon map can store an arbitrary number of properties for each photon, depending on available memory of the computer. It could store then, for example, the direction of the incoming rays and use these to compute areas of high interest for extra attention by the rendering ray tracer. Photon maps are created in a pre-processing step and can be parameterised for specific purposes. For example, photon maps could be created, but by sampling only those paths which intersect objects that cast caustics. This effect requires a high number of rays to achieve good quality.

Dynamic Scenes

Ray Tracing is no more or less amenable to dynamic scenes than the painters algorithm. A naïve implementation of a ray tracer supports as much variation in the geometric structure of the scene as the painters algorithm. For rigid transformations, the impact on efficiency is even less than that it is for the painters algorithm. This is because performing an intersection test between a ray and an object rigidly transformed, is equivalent to performing an intersection between the untransformed object and the ray transformed by the inverse transform. The vertex shader in the painters algorithm can be provided with a transform to apply to each vertex. In the same manner, each primitive in a ray traced scene can be provided with a transform for a ray before any intersection tests take place. This also makes instancing in ray tracers very efficient.

When animation and other dynamics in the scene are included, the application of acceleration structures is complicated. Some acceleration structures have been designed with animated geometry in mind, and are fast to update. Typically though the best performing acceleration structures such as KD-Trees take a considerable time to update. A further issue with acceleration structures which is exasperated by animation, is their dependence on scene structure. Grids, for example, have set cell sizes. If the cells are too large, then the number of intersection tests will be non-optimal. If they are too small update times will become impractical. Selecting the most efficient acceleration structure for a scene is difficult, but even more so when the scene content may change at any time [238].

2.4.2 Image-Based Rendering

Recall the rendering continuum from Section 2.3.1.1. The next stage along this from geometry-based rendering is image-based rendering. Unlike the techniques considered until now, image-based renderers store only minimal, if any, information about the geometric content of a scene. Image-based rendering is the construction of novel view of a scene, not by modelling the transport of light, but by sampling it. One advantage of image-based rendering is speed. In image-based renderers samples of the lighting in a scene are pre-computed outside the critical rendering loop. During rendering these samples are combined to approximate the lighting from a given perspective. They are either sampled from the real world with cameras or calculated from virtual worlds with techniques similar to ray tracing [200]. A simple but practical example of an image-based renderer would be an environment map applied to a sphere. The texture on the sphere would be sampled based on a viewpoint inside it, by casting a ray out towards the sphere surface. If the degrees of freedom of the viewpoint were limited to orientation only, all possible views of the environment would be contained within the map. Pre-computing samples reduces the computational load during rendering, but limits the dynamism of the scene. Any parameter of the scene which contributes to the appearance of the pre-computed effects cannot be changed without a possibly expensive re-computation stage, or the risk of introducing artefacts.

Figure 2.18: An environment map which can be applied to a sphere - it contains a full 360° view of the scene [124]

Image-based rendering exists on a spectrum with solutions trading off memory and on-line computational load for flexibility and quality. One of the purest implementations of an image-based renderer, meaning an implementation using little to no geometric information, is a light field renderer.

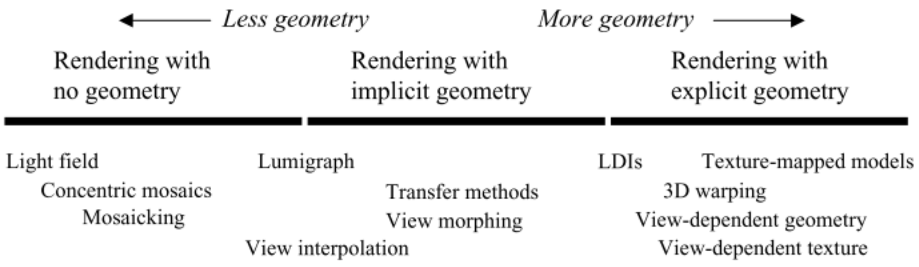


Figure 2.19: The continuum of image-based rendering [200]

2.4.3 Light Field Rendering

Light fields were introduced by Levoy & Hanrahan [114] who described them as samples of the plenoptic function. The plenoptic function [3] defines the radiance through a specific point in space. The solution is the total radiance of all the rays passing through this point, which may also be parametrised in terms of wavelength, time, and other factors.

Light fields are collections of samples of the solution to this function at discrete parametrisations. The goal of a light field renderer is to use these discrete samples to compute the continuous solution. As described by Slater [204], 'objects illuminate rays, and the rays are rendered'.

2.4.3.1 Parametrising light fields

The parametrisation of the light field defines the data structure in which the field is stored, and how samples are combined into novel viewpoints. For example, a renderer may be biased towards selecting samples with high spatial coherence or high angular coherence. This choice will affect the quality of the render and the artefacts that are introduced [87].

Light Slabs

The first parametrisation was introduced by Levoy & Hanrahan [114]. They noted that for many applications there are no occluders, and so the 6d plenoptic function can reduce to 4d. They parametrised rays using locations on two 2D planes, as shown in Figure 2.20.

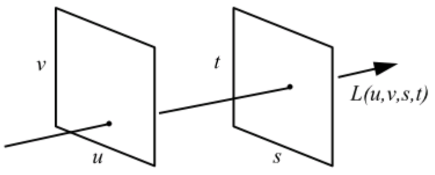


Figure 2.20: Illustration of a single ray in a light slab [114]

Schirmacher & Vogelgsang [191] created a generalised implementation of the light slab, with the near plane being a free-form surface. Gortler et al. [67] extended the parametrisation, providing each point on the near plane a basis function to create the *Lumigraph*. The function defines the contribution of that grid point, abstracting the integration from the rendering process.

Focal Stacks

Isaksen et al. [87] considered each location on the near plane a pinhole camera, containing samples at different directions. From the camera parameters, a mapping can be constructed allowing the correct sample from each camera to be identified for an arbitrary focal plane (Figure 2.21). This parametrisation allows a renderer to easily alter the focal depth of the rendered image, allowing for attractive visual effects such as in Figure 2.22.

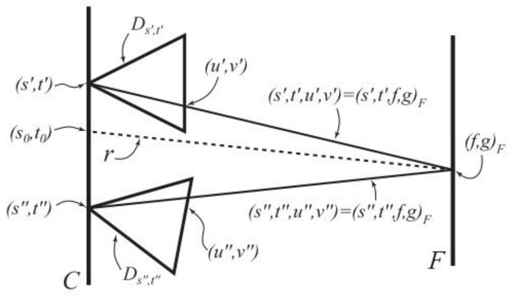


Figure 2.21: Given an arbitrary focal plane, and a location on this plane, the camera parameters can be used to identify the ray which intersects this point for each camera on the near plane [87]

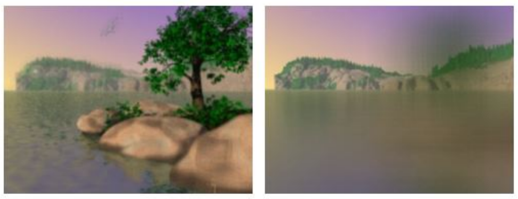


Figure 2.22: With a sufficiently large range of cameras on the near plane, a renderer can see through objects. No single camera in the light field will see the entire cliff face [87]

Spherical Light Fields

Ihm et al [82] introduced the spherical light field. It describes all the rays passing through a spherical hull bounding an object or scene (Figure 2.23).

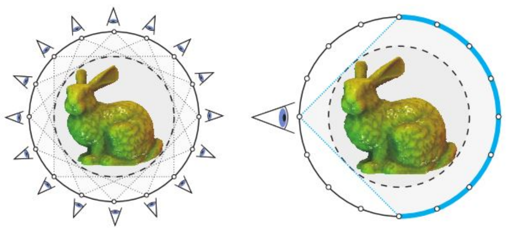


Figure 2.23: Illustration of a spherical light field expressed in 2D [227]

Concentric Mosaics

Shum & He [201] introduced the concentric mosaics parametrisation for inside-out and walkthrough applications. A concentric mosaic is constructed from multiple panoramas, captured in such a way that each vertical line provides a unique view of the scene. This is done using a linescan camera moving in a circle, but pointing tangential to it. The samples can then be rendered into a novel view as shown in Figure 2.24. The parametrisation was extended in [115] to include vertical line scans from different elevations, supporting vertical parallax. Birklbauer et al. [20] also used a cylindrical parametrisation but where the samples were normal to the concentric circles. Each location contained a number of samples of different orientations corresponding to different focal depths.

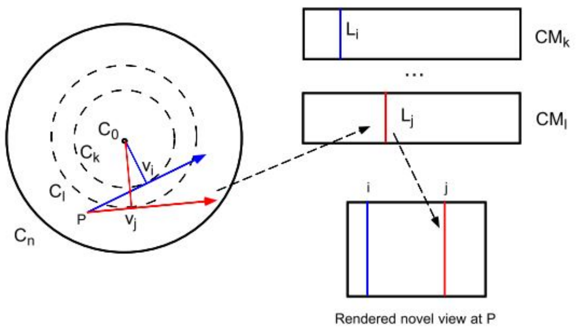


Figure 2.24: Rendering novel viewpoints with rays from concentric mosaics [201]

2.4.3.2 Virtual Light Fields

The light field parametrisations so far have assumed that light fields are captured from the real world. A number of light field cameras are available, from companies such as Lytro[8] and Raytrix[9]. Capturing light fields is not necessarily straightforward however. Rendering techniques assume a particular parametrisation, requiring potential transcoding of the field. Capturing uniformly spaced samples over a large area is logistically hard, especially with large rigs that may be required by, e.g. concentric mosaics.

Slater [204] introduced the Virtual Light Field (VLF), noting that light fields for VEs do not need to be constructed from a set of intermediate 2D images. A naive way to synthesize a VLF would be to simply move a camera through a scene rendered in real-time. This has a low barrier to entry, but would have the same limitations as the painters algorithm with regards to second order effects. High quality rendering

[8]https://www.lytro.com
[9]http://www.raytrix.de

techniques can be used in place of the real-time renderer but since each viewpoint must be rendered from scratch the time costs are prohibitive [144]. Slater's VLF is a data structure made up of multiple Parallel Sub-Fields (PSFs). A PSF is an array of *NxN* parallel rays, the rays all at some arbitrary but identical orientation around the center of an object or scene, analogous to an orthographic projection of the scene but with multiple intersection points for each ray. Khanna et al. [104], and Mortensen et al. [150], present a method for constructing virtual light fields by building an illumination network using a method similar to depth peeling.

2.4.3.3 Rendering Light Fields

The exact operations required to sample and render a light field depend upon the parametrisation. The first stage is identifying the parameters for the plenoptic function to solve. The parameters for a sample point on the image plane are computed analytically from the viewer position, perspective transform and position, just like in ray tracing [114].

This was described by Slater [204] as placing an eye into the scene. The eye, for example, could be a rectangular polygon within the volume of the light field, acting as a lens to focus and refract intersecting rays onto an image plane in order to support things like perspective projections. These rays would then be parametrised in terms of the light field and the continuous parameters for the plenoptic function computed for each texel on the image plane.

The second stage identifies the discrete samples with which to form the approximations of these continuous parameters. The approximations are formed by combining samples 'near' the discrete parametrisation.

Artefacts and Depth Corrected Rendering

The creators of the first light field renderer noted the requirement of very high sampling densities to avoid excessive blurriness [114]. While they describe it as a limitation of their method, this blurriness is a consequence of the smoothing process light field data undergoes as it is sampled from the real world by the camera, and then again as it is rendered from multiple samples. A more extreme artefact is ghosting which occurs due to low sampling densities and inappropriate focal planes. A good explanation of these artefacts is available in [118].

Assuming an arbitrary constant focal depth can result in severe artefacts [118]. A number of authors have used depth information, approximated from visual flow or available as a by-product of synthesis, to alleviate these.

Gortler et al. [67] combined a geometric proxy with their light slab. They use this proxy to compute the 3D location all sample points should pass through, and therefore new sampling locations on the near and far planes for each ray. In the original implementation the depth correction was performed dynamically, with the sampling rays being cast against proxy geometry. This means the light field renderer loses its deterministic characteristics. It should also be noted, as in [27], that a process which entails only looking for rays which intersect the same point can only approximate Lambertian surfaces with reasonable accuracy.

Buehler et al. [27] introduced a technique to render unstructured lumigraphs. They create a *camera blending field* which defines coefficients for each of the cameras in the lumigraph. This allows higher significance to be given to samples with smaller angular or Euclidean distances, for example.

Schirmacher & Vogelgsang had a similar approach in [191]. They created a surface, onto which the images from a set of unstructured cameras could be projected. By incorporating depth information from the original images into this mesh, they could subdivide & weight the source images and combine them appropriately.

The depth correction techniques considered so far have used geometric proxies in one way or another. While improving rendering quality, these make the rendering techniques non-deterministic and coupled to the complexity of the proxies.

Todt et al. [227] presented a depth-correct light field renderer using only the depth values for individual sample rays. They did this by estimating the depth of the rendering rays performing multiple iterative sampling stages per pixel. Each stage improved the depth estimation of the rendering ray leading to the selection of better sampling rays. They presented two depth estimation methods. The first used iterative depth refinement based on the depth values from the surrounding rays (Figure 2.25). The second used a ray-marching approach to search until the integrated depth value closely matched that of the sample rays (Figure 2.26).

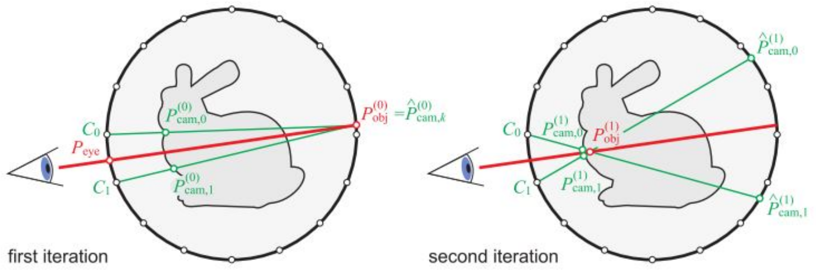

C_0
P_{eye}
C_1
$P^{(0)}_{\mathrm{cam},0}$
$P^{(0)}_{\mathrm{cam},1}$
$P^{(0)}_{\mathrm{obj}} = \hat{P}^{(0)}_{\mathrm{cam},k}$
first iteration
$\hat{P}^{(1)}_{\mathrm{cam},0}$
C_0
$P^{(1)}_{\mathrm{cam},0}$
$P^{(1)}_{\mathrm{obj}}$
C_1
$P^{(1)}_{\mathrm{cam},1}$
$\hat{P}^{(1)}_{\mathrm{cam},1}$
second iteration

Figure 2.25: Iterative depth refinement [227]

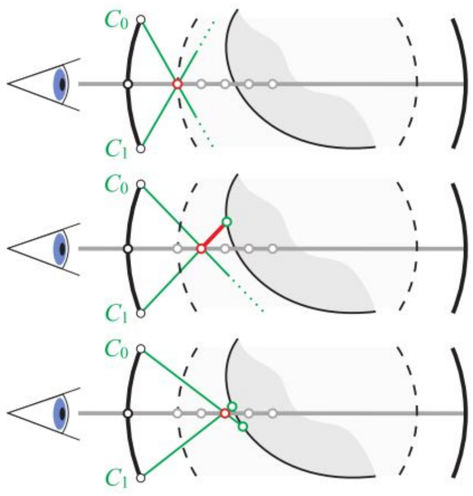


Figure 2.26: Ray marching depth estimation [227]

Light field density requirements

There is little discussion in the original light field publications, of how large or dense a light field should be for optimal rendering quality. Gortler et al. [67] suggested that since the object is assumed to lie near to the far plane, that this plane should match the final image resolution. Recall that in this parametrisation the closer the object is to the far plane, the fewer depth discrepancies there are (Figure 2.27). The resolution of the near plane in this case defines the angular resolution. For diffuse objects a low near plane resolution will be satisfactory while highly anisotropic surfaces will require a high resolution. Gortler et al. set the resolution of their near plane to ~1/8th of the far plane (512/64 & 128/16).

Lin & Shum [118] determined analytically the minimum number of images required for both the concentric mosaic and light slab parametrisations. They did

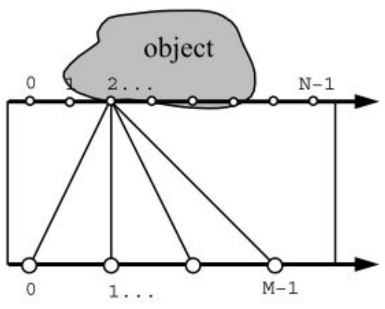


Figure 2.27: When the far plane and the surface of an object are spatially coherent, a grid point on this plane will always sample the same location on the object, with the near plane defining the perspective [67]

this by modelling the cause of the double image artefact and deriving expressions relating sample depth discrepancies to the sampling camera density.

2.4.4 Frameless Rendering

Rendering a frame typically takes longer than to draw it. Most renderers therefore compute an entire frame, calculating sets of pixels in sequence while writing the colour values to one of a set of frame buffers. Swapping between these frame buffers prevents the user seeing discontinuities as the image is rendered. Some renderers however break the traditional concept of a frame in order to achieve higher performance.

In Section 2.4.1.4 it was seen how ray-tracer implementations would distribute processing of rays, either in hardware or software, to take advantage of cache coherency. This involves collating computations that are likely to require access to the same areas of memory, so that this data can be cached and served locally, for example from the CPU cache. This cuts down on memory accesses which take time and lock out access to other threads.

Tile-based rendering involves splitting the frame into tiles such as was done in Odom et al.'s ray-traced CAVE [160]. However tile based renderers will typically exist on a single chip and break the frame in an effort to minimise expensive off-chip accesses. This removes the bottleneck introduced by the memory channel, which can be especially important in systems which share graphics memory, such as mobile phones.

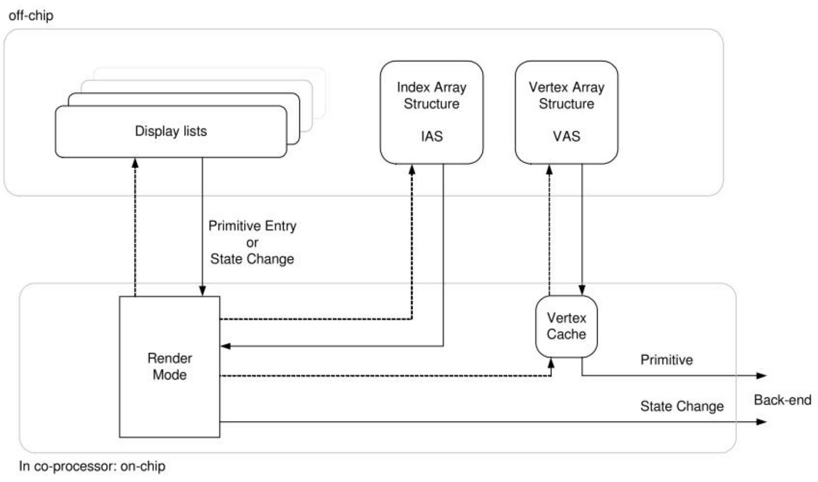


Figure 2.28: Tile based renderers attempt to remove the bottleneck of memory access by subdiving the frame into chunks manageable by a single chip's cache [29]

Figure 2.28 shows the data flow for a tile based renderer during rendering. Display Lists are constructed by identifying in which regions each primitive is present in. This is a relatively cheap operation compared to colour computation for each pixel. The display lists are used to cache locally the pertinent information required to render each primitive in the display list, such as vertex location and material properties. These local caches can then be used to render the tile quickly and update the shared frame buffer [29].

Tile-based rendering does lend itself to parallelism, though it is more commonly used to optimise memory access. Simply rendering tiles in sequence does not allow the frame buffer to be dispensed with as discontinuities may be noticeable along the large borders of the tiles. In [41], Dayal et al., present a frameless tile-based ray-tracing renderer which dynamically controls the tiling in order to target the ray-sampling into regions of high importance. In their implementation, an *adaptive sampler* identifies regions of the frame with large spatio and temporal colour changes. They cache samples over a period of time allowing the identification of regions with high detail such as edges, but also those in motion. The sampler directs the *adaptive re-constructor*, which uses ray-tracing and the cache of previous samples to compute new samples. It does this by tracing new rays, and re-projecting existing samples as a further optimisation.

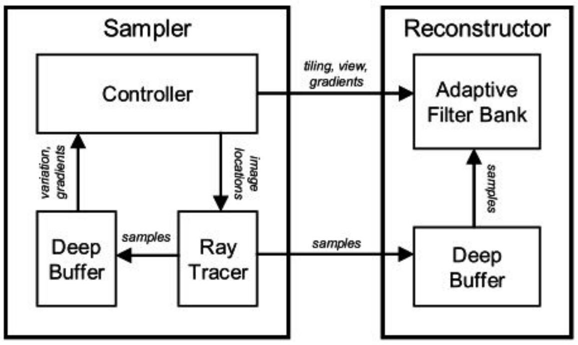


Figure 2.29: Dayal et al. intelligently directed a ray-tracer to re-sample tiles with high detail, in space or time [41]

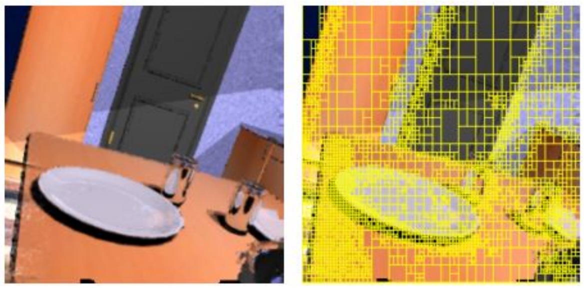


Figure 2.30: An example of the tile based segmentation performed by the adaptive sampler [41]

Figures 2.29 shows the structure of Dayal et al.'s renderer. Figure 2.30 shows an example of the frame segmented into tiles. The segmentation in Figure 2.30 is achieved by merging and segmenting tiles within the frame. This is done based on the information (amount of variation in colour) in samples contained within that tile. The adapative sampler will segment the frame so that the variations within each tile are roughly equal over the whole image. This results in tiles over areas of high detail tending to be smaller. The tiles to sample next are selected randomly. The areas with high detail or velocity have a higher change of being re-constructed as they have a higher tile count.

Dayal et al. implemented their adaptive frameless renderer on an NVidia 6800 Ultra GPU and acheived a "framerate" of 20 Hz at a resolution of 256×256. The figure of 20 Hz refers to the rate at which the renderer could re-construct approximately 400,000 samples. This figure has little meaning though as there is no reason to assume that higher sample rates will result in lower latency or better image quality as this is highly dependent on the scene. The authors instead pre-render a short animation to use as the ground-truth. They then have other renders attempt to re-produce this in real-time. The resulting error rates provide a good characterisation of the renderers in terms of image quality and latency & jitter. The adaptive frameless renderer presented shows error rates considerably lower than other frameless and frame based renderers.

Bergman et al. [19] were one of the first to describe an adaptive refinement algorithm. They introduced the concept of a 'golden thread'. The golden thread is a single step that as it is executed results in an ever higher fidelity image. Of course spatio-temporal driven sampling such as Dayal et al.'s is not the only option for adaptive refinement. Qu et al. [179] detected missing fragments in images after warps of voxel renders. Debattista et al. [42] and Chalmers et al. [35] used models of human perception to selectively guide rendering resources to the most impactful areas of the image, enabling high fidelity rendering for virtual reality at rates otherwise unachievable.

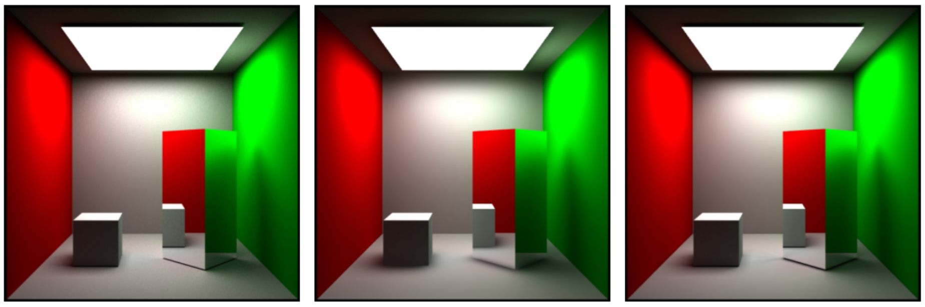
There are a number of alternative algorithms to solve the rendering equation. Ray-tracing typically has higher quality than the painters algorithm as more complex second-order effects can be simulated with it. It is in theory frame-less but in practice is far slower than existing pipelines due to the computational requirements of exploring the scene. The biggest impediment of these for real-time rendering is their non-deterministic nature. From the above previous works, it is clear image-based implementations typically have higher determinism and simpler local loops. Combining these approaches is a promising start for building low latency renderers, and we explore this in Chapter 4.

2.5 Specialised Rendering Hardware

2.5.1 Hardware Accelerated Light Field Renderers

Hardware accelerated light field renderers have been implemented, using both GPUs and bespoke hardware. In [150], Mortensen et al continued the work of Khanna et al and implemented a VLF synthesizer & renderer on a GPU. The authors used the same data structure and propagation process as described in Section 2.4.3.2. Rendering was done by re-sampling the directionally dependent radiance stored in the PSF tiles. To do this, the camera is placed within a unit sphere and the sphere rendered in false colour. This sphere is the same one used to describe the directions of each PSF. That is, each vertex defines a PSF direction. When triangles of this sphere are projected onto the image plane, they can be used to define the closest PSFs (direction wise) for each pixel. This is because each vertex describes a discrete direction, and during rasterisation the direction identifiers from the three corners of the triangle are interpolated across an area of the image plane. The value at any pixel then determines the weights for the PSFs surrounding it. Recall that each PSF contains sets of surfaces which intersect its various rays. Before interpolation can take place, the correct surfaces from the PSFs must be identified. To do this, the scene is rendered in false colour again, this time returning surface identifiers for each pixel. Another false colour pass identifies the position of each pixel in world coordinates. These coordinates are used to identify the exact rays in each PSFs with which to look up the luminances. The luminances can then be interpolated using the coefficients computed in the first pass.

For this implementation the rendering time is not independent of total scene geometry, but unlike other real time global illumination approximations it is independent of the viewpoint. The frame rates achieved were all between 121 and 124 fps for scenes with between 19 and ~9k faces. Mortensen et al also noted that their method of propagation outperformed many off-line ray-casting based global illumination techniques.



(a) Rendered with PBRT. (b) Rendered with VLF-GPU (low quality). (c) Rendered with VLF-GPU (high quality).

Figure 2.31: Comparison of the quality of a real-time hardware accelerated VLF renderer with an offline ray tracing algorithm [150]

Regan et al. [181] constructed a hardware accelerated lightfield renderer with the explicit goal of achieving low latency. Their system was designed for "fish tank" VR. That is, the user observes the virtual world via a display, but receives motion parallax cues, as if they are looking into a true 3D volume.

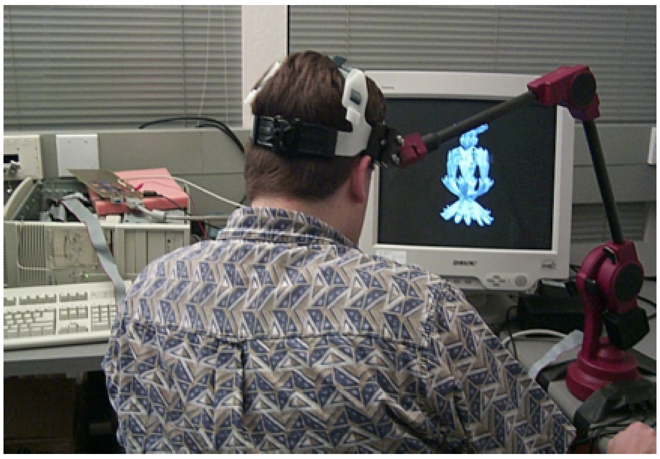


Figure 2.32: An example of fish-tank VR [181]

The authors did not state whether they captured the light fields or rendered them. They did however use Levoy & Hanrahans light slab representation. Their light slabs were assembled from 128 512x512 gray-scale images. The light field renderer only supported 1 degree of freedom, that of position in the horizontal axis. That is there were no vertical parallax cues. This is due to the memory limitations of the hardware (32 MB). The architecture of their renderer is shown in Figure 2.33.

The Raster Generator is responsible for maintaining the point in the frame and

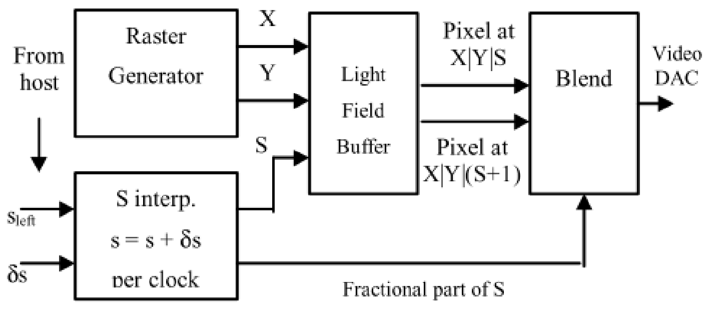


Figure 2.33: Block diagram of the architecture of a real-time light field renderer implemented on an FPGA [181]

runs continuously. The host computer forwards tracker data to the renderer in the form of the horizontal axis position change δs. On each clock cycle this is integrated with s and the current position is maintained within the renderer itself. The current position in image space is used to sample the light field in the Light Field Buffer. Two samples are returned, at s and $s+1$, and then these are blended depending on how far between them the user is in the Blend function. The tracker in this case is a rotary encoder which can be sampled at very high rates by the renderer and so position samples are very likely to have higher resolution than the light field sample intervals. Regan et al implemented their renderer on an Programmable And Reconfigurable Tool Set (PARTS) board. This was to avoid the output buffer of existing graphics systems and keep latency low. The PARTS board is somewhat like a precursor to a Dataflow Engine (DFE). It is an FPGA development platform consisting of Virtex 4028 FPGAs running at 15MHz which could be placed in a PCI slot.

The authors measured the latency of their system at 200us. They used it to investigate thresholds of latency perception. They found that users could begin to detect latencies around 15 ms.

2.5.2 Address Recalculation Pipeline

Regan & Pose created the Address Recalculation Pipeline, which decoupled the user's head orientation from the rendering process [180]. It did this by continually rendering a scene into a cubic environment map. This was then sampled at a high rate, each sample using the latest orientation tracking data. By decoupling the generation of the final image from the slower traditional rendering process, Regan & Pose could minimise apparent latency. Using image composition, they could also combine parts of the scene rendered at different rates, or using different techniques or even hardware. Regan & Pose's implementation was designed to run with a pixel clock of 25.2 MHz, each of the five pipeline stages executing within 40 ns. If each reported stage was atomic, the total latency would be 200 ns.

2.5.3 Warper Board

Vicenzi et al. [235] used a *warper board* to compensate for latency in HMDs. This was a dedicated device with a hardware implementation of a set of static and dynamic image warping algorithms. The board would perform image distortion for the HMD geometry, in addition to extrapolating the image to create a new view based on the latest tracking data.

By using a predictive filter such as a Kalman Filter, the authors could reduce the apparent latency of their HMD. They tested a number of predictors, with and without the warper board and found that the warper board combined with a Kalman Filter provided the best performance - a visual registration error of ~0.26°.

2.5.4 Warp Engine

Popescu et al. [178] created the WarpEngine. This is similar to Vicenzi et al.'s WarperBoard but operates on images with depth information. Reference images are subdivided into tiles which form the basic rendering primitives. The tiles are then warped and interpolated into screen space. The interpolation is performed using a *warp buffer* - an oversampling of the screen space with 2x2 depth pixels interpolated per pixel in the final frame.

Popescu et al. use multiple ASICs running in parallel to perform tile based rendering. This was originally because a single ASIC did not have sufficient processing power or memory. However, it has the side effect of reducing latency should the regions visited later in the scan-out be able to be fed with updated tracker data, or, ideally drive display like a Digital Micro-mirror Device (DMD) that is not line-scan based. As the same operations are performed on each tile primitive, the WarpEngine design maps well to SIMD based architectures so may now work well on a GPU (GPUs at the time were not powerful enough to support the implementation).

The authors did not measure the motion to photon latency of their system, however they did estimate it took approximately 1872 cycles to perform a 3D image warp for all samples in a tile. With a hardware implementation running at 300 MHz, the rendering latency would be on the order of microseconds [133].

2.5.5 High Performance Touch Prototype

Jota et al. [98] and Ng et al. [155] created the 'High Performance Touch' prototype. This apparatus was designed to facilitate experiments with direct touch interfaces. The apparatus approximated a hardware accelerated pathway proposed by the authors that could be incorporated into devices such as smartphones. This channel would receive user input and drive the GUI of the device independently of the application that created the control. The application would process the input in the background and update the GUI a few ms later, but the user would have the perception that the device is responding instantly.

The apparatus is based on a low-latency touchscreen connected directly to a DMD projector. An Field Programmable Gate Array (FPGA) development kit receives input directly from the touch panel. A number of basic controls, such as sliders, are implemented in the FPGA hardware design, allowing it to respond as the proposed parallel channel above would. The system is managed at a high level by a separate PC. The latency of the system is configurable, with a base latency of 1 ms.

Jota et al. [99] used this apparatus to perform investigations into user interaction using Fitts' Law-style tests. A linear regression showed a significant effect of latency on throughput (i.e. user performance), though a pair-wise comparison showed no significant affect of latency between the 1ms and 10ms conditions. The authors also investigated the consciously perceptible latency, and found users could perceive as little as 20ms of latency between them placing their finger on the touch panel and the system responding. During direct interaction tasks participants can 'detect' as little as 2-3ms, though this is believed by the authors to be them detecting the disparity/registration error of the display, rather than the latency itself.

2.5.6 DMD Optical See Through Display

Lincoln et al. [119] used a DMD to create a prototype AR system. Their implementation contained the start of what should become a system of cascaded image warping functions, each running at a higher rate. The design is based on the observation that the magnitude of the error introduced in a warping stage, corresponds to the magnitude of the warp. Starting with a frame rendered from a traditional GPU, they chain multiple warping stages, each one simpler than the one before it, but operating at a higher rate. The higher speeds mean the corrections applied by the cruder stages are less significant and therefore less visible, but the perceptual latency is still only as high as the final stage [263]. The system uses dithering to create grayscale images from the monochrome DMD. The image warping functionality is embedded in the control loop that drives the dithering. The warping functionality supports accounting for rotations in two axis. It does this by moving a sliding window across the image. Monochrome frames are displayed at 16 kHz allowing the system to produce a visible response to user input within 80 microseconds.

In the current implementation, the authors precompute a frame and write it into the FPGA, rather than feed it a stream of semi-distorted images as would be the case in the final implementation. To reduce the latency as much as possible, they use a mechanical tracking system.

There have been a number of specialised rendering systems built to optimise for latency. It is clear from these that the constraints required to achieve the high performance of the local rendering loop make them unsuitable for shouldering the entire rendering problem. Most augment existing systems or otherwise accept severe constraints as part of their nature as proof-of-concepts. We also note that a number of these systems such as Regan et al.'s were limited only by the hardware of the time and that we are able to re-use their principles directly, removing constraints solely with our more capable hardware.

Part I

Low Latency Renderers

Chapter 3

Dataflow Computing and Rendering

To prototype the alternative dataflow rendering algorithms we used a Maxeler DFE, a spatial computing platform. This section provides a review of spatial computing, and the capabilities of the platform. We focus on the characteristics of dataflow computing that are advantageous to fast rendering local loops, specifically its determinism. Likewise, we review the characteristics of the Maxeler platforms which have the most influence on the implementations, such as memory and IO. Finally, we review a proof-of-concept just-in-time renderer that draws 2D sprites.

3.1 Spatial Computing

In traditional CPU architectures, a single piece of multifunctional silicon executes all of the operations in an algorithm. It executes the algorithm on one unit of data, with the operations being laid out in time. Subsequent units of data wait until the processor has executed all operations, before they are processed in turn. In dataflow computing, the algorithms are laid out in space. Algorithms are expressed as a sequence of operations which are arranged into dataflow graphs. Data to be transformed by the algorithm is streamed into the graph and moves between the operations being transformed at each stage until the algorithm is complete. Each operation has a dedicated piece of silicon. All operations execute with true parallelism. The computation time of the dataflow graph for a single unit of data then is the same as that on a CPU, but the dataflow graph is executing the algorithm on many units of data simultaneously. The throughput then is far higher than a time-shared processor

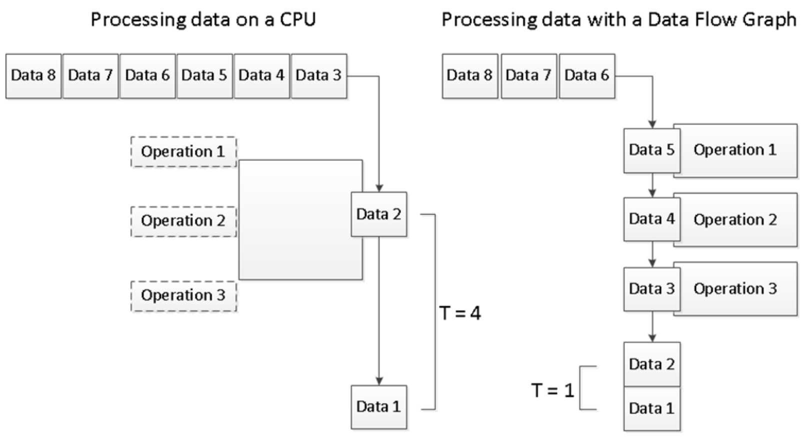


Figure 3.1: Illustration of how true-parallel processors achieve higher throughput than time-shared processors

[170]. This is illustrated in Figure 3.1.

All common architectures can execute dataflow graphs, including CPUs and GPUs, as they are simply a sequence of operations. To get the benefit of expressing the algorithm in this way though, a spatial computing platform is required which allows for true-parallelism.

3.2 Maxeler Dataflow Engines

Maxeler DFEs are processing cards which execute dataflow computations. Users describe their dataflow graph in Java using Maxeler's toolchain (MaxCompiler). The toolchain compiles the algorithm into a form which can be loaded onto a DFE. When the design is loaded, the DFE is a computer dedicated to executing that algorithm. MaxCompiler & the runtime services (MaxelerOS) abstract away the implementation details of the DFE. For example a user can compile their design and then run it on a card installed locally, or a set of cards installed remotely in a data centre, with only slight modifications to the application running on the host PC, and no modification to the dataflow graph [138]. The current technology that DFEs are based on are FPGAs. Any spatial computing technology could be used in future generations however. In this section we review design considerations and DFE resources that are relevant to any dataflow algorithm designer. We also review how the current generation of DFEs are constructed. This is relevant, as to minimise latency a rendering algorithm must be tightly coupled to the hardware driving the display. This is because the current generation of displays operate at fixed frame rates, and display data which is computed at a rate which leads or lags that of the display will introduce artefacts and additional latency. This has been recognised by industry and standards have been announced for variable rate displays, but they have not yet achieved widespread adoption [233].

The computational power of a DFE is derived from an FPGA. It is expected the reader has some awareness of what FPGAs are and where they are used. FPGAs are ICs, which can be reconfigured to implement any arbitrary logic circuit that fits within the device. Circuits are formed in the FPGA fabric, which is where the combinatorial logic is implemented, and interconnections between this and the resources of the chip are formed. In addition to the components that form the fabric, modern FPGAs include hardware dedicated to specific functionality. The structure of such an FPGA, the Altera Stratix V, is shown in Figure 3.2. The fabric is where the user design is implemented. Interspersed throughout this is dedicated hardware. The DSP Blocks, for example, implement common digital signal processing functions such as fixed point multipliers and accumulators. The M20K Blocks are memories. The blocks to the left and right edges are hardware implementations of the lower layers of common communication protocols such as PCIe and 10G Ethernet. The blocks at the top and bottom implement circuitry for communicating with DRAM memories and general purpose IOs [8].

The dataflow graph constructed by Maxeler's toolchain shall have to be connected to the resources described, to form the physical layer of a channel which will drive a display. To minimise latency, this channel should be directly between the dataflow graph and the display. We will not discuss the implementation of FPGAs further, other than to say that FPGA fabrics do have maximum supported clock rates. These are a function of device design and are specified by the manufacturer. They are typically lower than CPUs, even low power CPUs in microcontrollers, and ASIC implementations of the same design. For example, the Stratix V core fabric can theoretically run at ~800 MHz; the actual speed depending on limiting factors, such

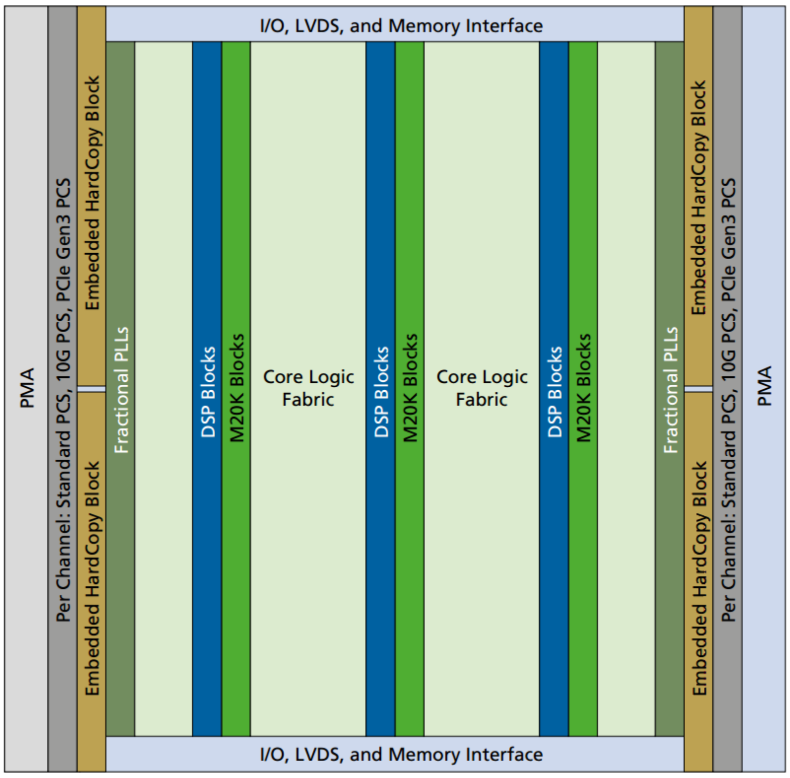


Figure 3.2: Diagram of the physical layout of a Stratix V FPGA [8]

as the length of individual routes, within a design [8]. A detailed review of FPGA technology is available in [139].

Within the dataflow graph, MaxCompiler supports common logical operators and control flow mechanisms such as ternary operators and switch statements. There is extensive support for mathematical operations on arbitrary precision numerical types, from simple operations to numerical methods such as finite difference for approximating derivatives [138, 171].

To be of any use, the edges of a dataflow graph must communicate with external devices to send and receive data. What forms of communication are available depend on the resources of the DFE. Some DFEs support specialised data sources such as GPS and high precision clocks, but the most common interconnects are MaxRing, Ethernet and PCIe. In addition we have implemented a display interconnect allowing us to drive any DVI receiver. These resources are implemented using a combination of Hard IP on the FPGA, physical resources on the DFE, and Soft IP provided by the Maxeler toolchain, which also provides an interface to the dataflow graph.

3.2.1 Maxeler DFE Interconnects

3.2.1.1 MaxRing and Ethernet

MaxRing and Ethernet are high speed serial interconnects which allow DFEs to connect to other devices. Unlike PCIe which connects only to the host, MaxRing and Ethernet allow a DFE to operate autonomously - receiving, processing and transmitting data independently once the dataflow algorithm has been loaded. The hardware implementation of the communication stack for both MaxRing and Ethernet make transmission between devices deterministic as long as the channel is not interrupted. These connections can be used to spread a design across multiple devices, increasing memory size and bandwidth, or computational power [136]. The latency through the stack on the DFE is also very small, in the order of nanoseconds.

When using these interconnects there are two considerations. The first is straightforward, and that is that the bandwidth of the links are limited, and the designed data rate including overhead must not exceed this. The second is less obvious but significant for our application. Modern high bandwidth connections such as these are based on high-speed serial protocols which are asynchronous or plesiochronous. With these channels, the clocks at the transmitter and receiver are expected to differ. This is due to the practicalities of transmitting such a high speed clock between devices. The clocks for the channel are synthesized independently at each device, with the receiver attempting to match its clock to that recovered from the data. PCIe for example expects the clocks at the transmitter and receiver to differ by up to 600 ppm [202]. These differences require control symbols to be inserted into the data stream for clock detection & phase alignment. Protocols also include space or idle symbols which are dropped by the receiver. These are used in rate matching implementations where the transmitter and receiver should operate at nominally the same rate, but the transmitter may be operating slightly faster [70, 7]. Ethernet, for example, uses this approach. These channels typically support packet based transmission protocols such as Ethernet, USB and DisplayPort, which by nature support discontinuities in the data stream. In our application though an image generator must maintain synchronization between data sources, and must operate continuously, indefinitely. As a result data sources external to our rendering algorithm must operate at a higher rate than our pixel pipeline. This is to prevent the pipeline being starved, since it cannot be halted to wait for data without introducing visual artefacts. High level flow control will be required as well, to prevent overflow due to the higher rate of the data sources. The responsiveness of the flow control implementation will determine how much data must be pre-computed, and therefore the additional latency in our pipeline.

MaxRing uses the Aurora protocol [255]. This is a high speed serial simplex or full duplex protocol designed and made open by Xilinx. Being open, it is one of the few protocols that can be used for chip-to-chip communication between FPGA vendors. Aurora is a frame-based protocol with control messages and frames existing at the same level. These control messages include native flow control messages which can be used to pause a transmitter indefinitely or for a fixed number of cycles. When paused, the transmitter sends idle messages instead of frame data [183, 255]. Ethernet defines spacing symbols (known as IDLE as well), but not for-rate matching. This function is fulfilled by the Inter-Packet Gap. This is the time that transmitters must remain idle between frames, and originates from when Ethernet was a bus-based protocol and transmitters needed to remain idle for a proportion of the time to allow others the chance to transmit. The Inter-Packet Gap does however provide an opportunity for a receiver to process the last received frame [203]. The IDLE symbol in Ethernet is used instead to keep the signalling system alive when there is no data to transmit. Ethernet also includes flow control in the form of PAUSE frames [101].

3.2.1.2 PCIe

PCIe is a high speed point-to-point based interconnection fabric. Current generation DFEs have PCIe connections through which they communicate with a host. MaxelerOS provides a low latency API. This uses polling to minimise the latency when communicating with a DFE [136].

Typical interconnect latencies are in the order of hundreds of nanoseconds, so PCIe latency should be relatively inconsequential [177]. We verified the total end-to-end latency of the system regardless. Using high speed cameras and a modified keyboard, we show the latency through the OS input system, PCIe and DFE is under 1 ms.

3.3 Control Flow in Spatial Computers

Control flow for branching is supported in dataflow graphs. Much like in early GPUs though it can be inefficient since both pathways are executed but only one result is used. Control flow is more common in the form of controlling access to resources. For example, disabling transmission of network data while frames are assembled. In addition to comparison operators, spatial computers provide the concept of time in the form of counters [138]. Looping and recursion are possible in dataflow computing and can be implemented very efficiently in terms of hardware resources, but they can impact the data rate severely. Recursion is implemented in much the same way as looping; one edge of the data flow graph is attached further up the graph forming a feedback loop, known as a cycle. In a dataflow graph the operations are not implemented as a continuous network of combinatorial logic. On each clock tick the data progresses through the pipeline one stage at a time. So, for example, an algorithm consisting of 8 operations would take 8 ticks to complete. Knowing the distance in operations between two edges of the graph, allows a stage to incorporate previous results into its operations. Figure 3.3 illustrates this concept for the row sum problem, where each new data value is added to the sum of those before it.

It is important to remember that what appear to be atomic operations such as additions and multiplications may be implemented in multiple stages. This has two implications, the first is that pipeline must be tapped at the correct point, and where this is may not be obvious. The second is that the result of this operation will be delayed by the number of stages in time. Consider Figure 3.4, which is a more detailed illustration of the row sum problem. The addition operation is split into four pipelined stages. As can be seen though, on each clock cycle a new value is read from the input stream, even when the result of the previous operation as not reached the end of pipeline. It can be seen that due to pipeline delay, by the time N_1 reaches the

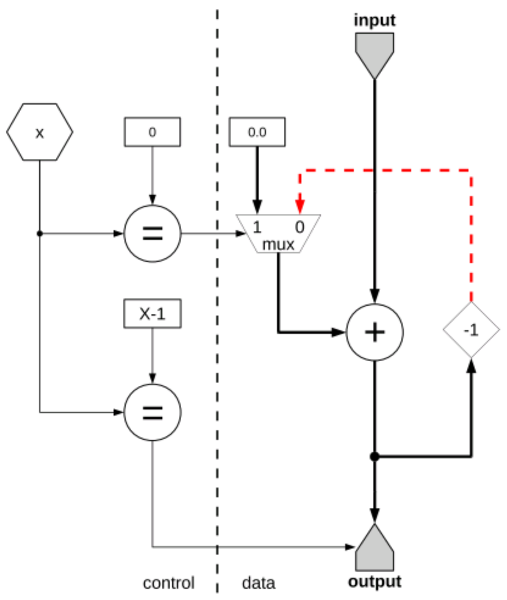


Figure 3.3: Illustration of how cycles may be used to implement loops in dataflow graphs [135]

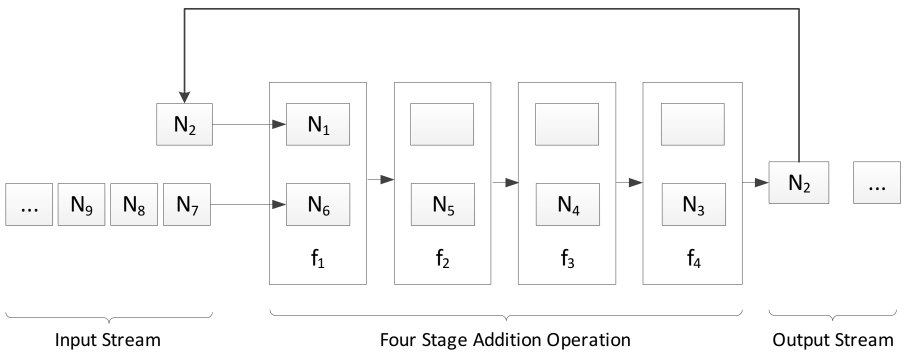


Figure 3.4: Illustration of how ostensibly atomic operations may be split up in practice, confounding the design of a dataflow cycle

input again, three other values have already been read, meaning N_1 will be summed with N_6, N_2 with N_7 and so on. For the result to be mathematically correct, N_1 should be summed with N_2. For this to happen, the input stream must be disabled after every new read, until the addition pipeline is flushed. This destroys the true-parallelism of the pipeline and reduces the throughput by a factor of four. Similar issues will be encountered with recursive algorithms, where data is passed through the same pipeline stages repeatedly. So long as there is no cyclic data flow however the loops may be unrolled, using more logic but dispensing with flow control [135].

3.4 Memory in a DFE

The main memory resources of a DFE are in the form of DRAM, specifically DDR3. DRAM modules are connected to the FPGA, and memory controllers are instantiated by MaxCompiler within the design to drive these. Currently, the memory controller concatenates all the DRAM modules, and so they appear as one very wide memory. The address width depends on the number of modules. The interface that is presented to the dataflow graph supports many streams of commands and data, operating in parallel. Internally access to the memory is multiplexed between these. The memory controller has advanced arbitration abilities which select which stream receives control of the memory, based on rates the various streams are operating at [247]. The controller is designed to prevent data starvation, but makes individual access non-deterministic.

The memory controller operates in burst mode, reading multiple words per access. Burst accesses provide high efficiency, but further, they allow the memory to operate at much higher rates than the FPGA fabric supports. For example, the FPGA fabric may run at 200 MHz and the DRAMs at 400 MHz. Externally four words would be read from a 64 bit DIMM, but internally these would appear as two words of 128 bits each; two sequential accesses to the memory being concatenated into one long word on the FPGA, halving the speed of the bus. The current generation of DFEs have 6 DIMMs which are presented as a single memory with a width of 1536 bits. That is 256 bits per DIMM per internal cycle. The memory may operate with a burst size of four (4 external cycles, 1 internal cycle, 1536 bits) or eight (8 external cycles, 2 internal cycles, 3072 bits between two words) [137, 247].

In theory, a burst size of four would result in one internal cycle per operation. In practice the latency of the memory controller is relatively high, up to 60 cycles from transmitting the command to actually receiving data (personal communication). Further, it cannot be predicted with perfect accuracy due to the DRAM requirement for refresh cycles and its non-trivial addressing. DRAM is addressed by banks, rows and columns. These three components together identify a unique bit. Internally DRAM memories are partitioned into arrays. The bits from each array (addressed by the same bank, row and column numbers) are concatenated to form the data word. These addressing elements are set independently in distinct cycles. This is a consequence of the addressing circuitry. It does allow the elements to share physical pins, but means it takes longer to access a series of sparse addresses than sequential ones. Pipelining in DRAM memories can alleviate some of the performance loss, but cannot avoid the necessity of setting multiple address elements [103]. When an algorithm will make many small random accesses to a smaller dataset, there is a way to improve performance further. This involves duplicating the dataset in each bank of a memory, and then distributing individual random accesses to each bank in turn. This takes advantage of the fact that each bank in a memory has its own addressing circuitry. If enough addresses can be computed ahead of time to feed all the banks, one bank can be preparing access to one address, while the other prepares access to the second address, and so on. This way, a new word can be received on every tick.

If an algorithm requires a uninterrupted stream of random access data then, there are two considerations. The addresses must be computed a sufficient number of cycles in advance to overcome the latency of the memory controller or the reads will stall and the memory will not operate at maximum efficiency. The memory controller must also operate at a sufficiently higher rate than the algorithm, in order to complete the addressing procedure, receive a full burst of data, and perform any refresh cycles, all within one tick of the receiving algorithm. It is possible to trade-off memory space for speed by duplicating data.

MaxCompiler provides functionality beyond driving the memory. For example it is capable of determining when multiple parts of a design read the same address, and fan-out the data from one read to these. It can also generate logic to ensure predictable behaviour when, for example, reading and writing a single address within one cycle. In addition to the large DRAM memories, MaxCompiler provides access to FMEM (Fast Memory). FMEM is on-chip memory, and a few MBytes are available. FMEM is implemented using discrete and independent resources spread over the physical surface of the chip (the M20K blocks in Figure 1). Therefore FMEM has truly parallel access, so long as the accesses are not to the same location in memory. The FMEM is implemented as SRAM; this has a much lower latency than DRAM, making FMEM suitable for local caching [138].

3.5 Real-time Rendering on a DFE

In this section we review a number of algorithms presented in Section 2.3 from the point of view of implementation on a dataflow computer.

3.5.1 Painters algorithm

The painters algorithm maps very well to dataflow processing, for the same reasons it maps well to stream processing. Dataflow architectures will not however help it overcome its biggest limitation in VR - that of the inability to update mid-frame. The ability for a DFE to perform computations on incoming memory streams ‘for free’ does offer some opportunities for improving how this algorithm has been implemented in hardware. For example, an acceleration structure could be constructed as the scene geometry is loaded, allowing culling to be performed on the DFE. The DFE could contain scenes much larger than current GPUs. Effects such as shadows could be implemented with true parallelism. Much more complex BRDFs could be supported, and they could support more lights than are typically available, with only a trivial increase in rendering time (although at a significant cost in FPGA fabric space). Translucent fragments could be blended efficiently with dedicated hardware, or this could be brute forced with a 3D colour buffer in the very large LMem. Further, image filters could be applied at scan-out for free. It is unlikely these gains are worth more than the flexibility that GPU computational engines provide however. In every new generation of GPUs more and more dedicated functions are subsumed by the multiprocessor cores.

3.5.2 Ray tracing

Ray tracing is not the most amenable method for real-time rendering to begin with. The computational requirements increase with both scene complexity, image quality and final frame size. The render times also change depending on where in the scene the viewpoint is at any given time. The increase in throughput is the most attractive feature of a dataflow based ray-tracer. The quality of ray traced images, both in terms of artefacts and supported effects is typically determined by the number of rays cast. The advantage of using a dataflow architecture is that, like ray packets on a SIMD architecture, the additional cost in time of performing operations on multiple rays in parallel is nil. Unlike SIMD architectures however the dataflow units can differ significantly in what operations they perform making ray divergence less of an issue. The control flow requirements of ray tracing however are significant, as individual rays begin to take different paths through the scene geometry or acceleration structures, resulting in accesses to more disparate addresses. This is the biggest hurdle to a dataflow implementation, at least for real-time ray tracing. Much of the previous work on real-time ray tracers or ray tracers in general does not apply to implementation on a dataflow architecture. The mapping to GPUs is of little use since dataflow graphs have different capabilities to SIMD units. While tree structures may help reduce the total number of operations required to trace a ray, none of those examined so far make this number deterministic.

Dataflow computers are not suitable for implementing an entire ray tracer. This is because ray tracing algorithms make random memory accesses and parallel executions always diverge. However, dataflow computers could contribute to stages of the algorithm that could make use of the large amounts of localised memory. Caustic Professional built a ray-tracing accelerator card by reformulating the ray-tracing problem as a database problem [78, 39]. Noting that the majority of the algorithm's time is consumed performing intersection tests, they designed hardware to take advantage of a bespoke acceleration structure to trace rays at high speed. The complex lighting calculations were then offloaded to the CPU. In [106], Kim et al demonstrated a bespoke collision detection engine based on a Virtex II which could perform ray-triangle intersection tests. It outperformed both a GPU and CPU, the CPU by a factor of 70.

Consider Figure 3.5, which is the design for a naïve ray-intersection accelerator. This accelerator could perform $f \cdot n \cdot m$ intersection tests per second. Assume we are operating on a MAX4 DFE, we could say n is 5, as this DFE has a memory width of 1536 bit and a single precision triangle is 288 bit. Assume also that the memory runs at 433 MHz. Discounting 0.64% of the bandwidth for refresh cycles, we could clock the fabric of the FPGA at 215 MHz. Take *m* to be 256, based on the number of DSP multipliers available in the Stratix V, and required by the Moller's ray-triangle intersection algorithm [8, 106]. This design could then in theory execute 275 billion intersection tests per second. Clearly this design is not practical. Triangles for testing are assumed to be contiguous and where the sets begin are controlled at a coarse level by the CPU. It assumes the rays are coherent, or the CPU is taking a brute force approach by cycling through all the scene geometry on each iteration. It does serve to illustrate where spatial computing could benefit this problem however. The large memory bandwidth supplies triangles for testing in high volumes, with fan-out occurring for free in terms of time. The true-parallel nature of the implementation allows multiple intersection tests, on multiple rays, to take place simultaneously. Dataflow graphs could also implement some acceleration structures for free, for example partitioning triangles into a uniform quadtree could be done as triangles are loaded.

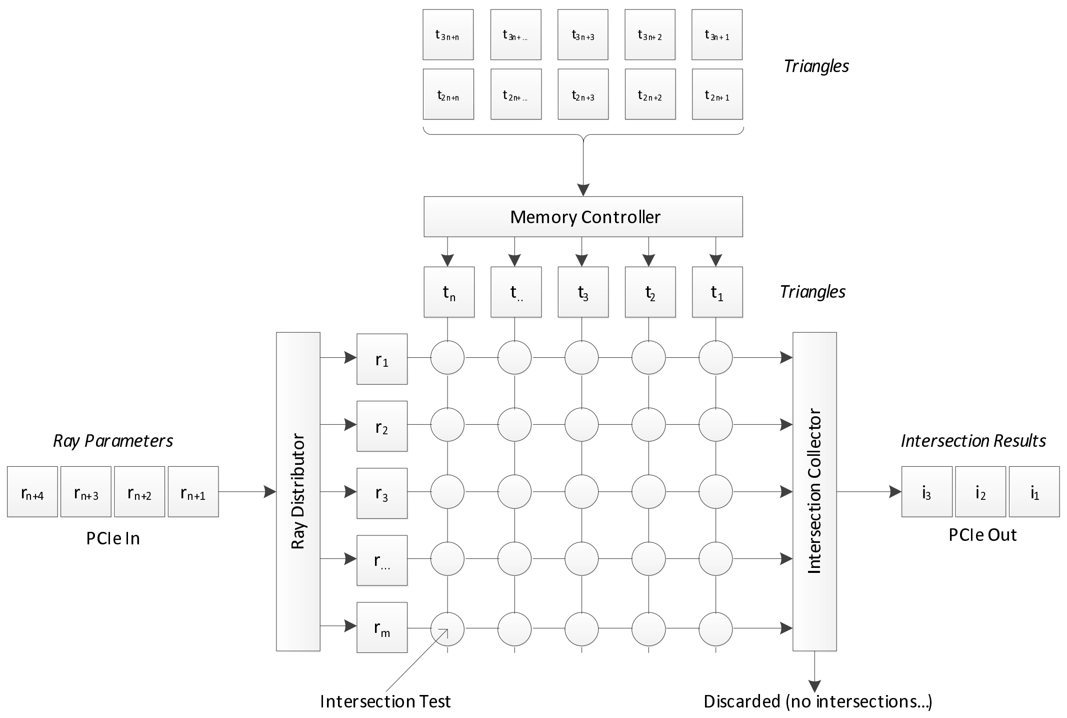


Figure 3.5: Architecture of a hypothetical ray tracing accelerator on a spatial computing platform

This would not be the first hardware accelerated ray-tracer. For example, Woop et al. [252] presented the design for a hardware accelerated ray-tracer prototyped on an FPGA. In 2013, Caustics Professional released a ray-tracing accelerator card [78]. Yongsheng et al. [258] also describe a hardware ray-tracer, but built with dataflow computing. Like Woop et al, theirs performs both intersection tests and shading in hardware.

3.5.3 Light field rendering

Light field rendering with its deterministic critical rendering loop and large memory requirements is well suited to implementation on our dataflow platform. That the rendering time is independent of scene complexity makes it especially suitable for VR. One of the big advantages of a DFE over a GPU or other co-processor is not only that there is a large amount of memory available, but that the dataflow graph has the opportunity to perform compression/decompression between the algorithm and the memory transparently. Decompressing light maps on the fly on a CPU or GPU would increase rendering time. It should be considered however that light field samples do not necessarily have high coherence. Selecting a data structure and compression technique which work well together and allow low latency random accesses would not be trivial. Once the samples have been retrieved though rendering process consists of a number of interpolation and filtering stages, all of which have very high locality and independence. The image quality is highly dependent on these operations, but selecting computationally intensive operations will not significantly increase rendering time as it would on a time-shared processor, but merely consume more space on the FPGA.

Being an entirely sample-based representation, one of the biggest issues with light fields for VR is the time it would take to update them. A high level of agency, or the sense of the user that they are affecting the world, is required for effective VR. The illumination in a light field rendered scene would be more realistic and reactive than one with baked maps. For example, the reflections and specular highlights would change immediately and realistically with the changing viewpoint. The increase in presence due the ability to alter ones own perception of the world may be significant owing to the more realistic response and reduced latency. The applications would be limited to those where interaction with the world beyond this is not important however. It would be impossible to present highly dynamic scenes unless some method of updating the light field in near real-time could be devised. Light field renderers do have the advantage the rendering process is independent of this. Birklbauer et al in [20] demonstrated such a system for rendering image stacks produced by scanning microscopes. It takes considerable time to render a perspective from these volumes, while rendering from a light field can be done very quickly. Birklbauer et al.'s implementation would render a new frame from a light field cache immediately on a change in perspective. The remaining time between perspective changes would be dedicated to updating this cache in the background.

3.5.3.1 Light field sizing and sampling

Current generation DFEs have 48 GB of memory available. It is prudent to calculate how much of this may be used by a typical light field for a VR experiment, as this will have implications for the design of an algorithm, such as the possibilities of memory space/bandwidth trade-offs. Lin & Shum presented a way to estimate the required number of samples for two common light-field parametrisations, and we begin with this. Lin & Shum's expression for the requisite camera spacing on the *uv* plane of a light slab is repeated in Equation 3.1 ([118]), along with an illustration of its parameters in Figure 3.6.

$$D_{max} = \delta(R+d) \cdot min\{\frac{A+d}{R-A}, \frac{B+d}{B-R}\} \tag{3.1}$$

We consider an environment that could be used to perform basic balance and locomotion tests. A cube with a volume 4 m^3, viewed via an HD HMD with a resolution of 1080 pixels and an FOV of 100^o. For simplicity we perform our calculations for one dimension and multiply it appropriately. If this scene had minimal depth discrepancies, e.g. 5cm, based on Lin & Shum's expression 30 camera positions would be required per dimension. We assume a camera resolution of 1080 to match the display, as per [67]. This results in a total light field size of 19

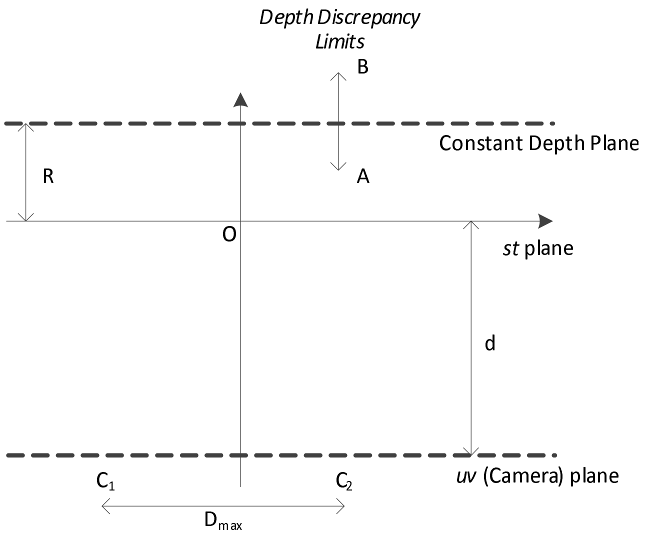


Figure 3.6: Illustration of the parameters of Lin & Shum's expression for the camera resolution requirements of the light slab parametrisation. This parametrisation assumes a constant focal depth across the entire object or scene, indicated by the *constant depth plane*.

GB for all six slabs making up the cube.

From Equation 3.1 we can see that increasing the variation in sample depth will increase the size of the light field. Increasing the depth variation to as little as 10 cm in either direction quadruples the light field size to 72 GB in the case above. Lin & Shum stated that they derived their expression based on minimising ghosting, and did not consider other artefacts. It can also be seen that the light field size decreases as angular resolution, given in degrees per pixel, increases. While minimising light field size, this will result in increased low pass sampling and therefore blurriness. The implications of considering only artefacts due to geometry can be seen when there is no variation in depth, and A & B in Equation 3.1 tend to zero. In this case D_{max} tends to infinity, i.e. the entire field can be expressed with a single camera position, completely removing any anisotropic features. Given the impact of depth variations on the light field size, it would be worth extending Lin & Shum's approach to calculate requisite resolutions for depth correct rendering techniques. If these techniques alleviate the need for such high sample densities, we could base the densities on the spatial frequencies of anisotropic surface features instead.

3.5.3.2 Non-uniform sampling

So far the light fields considered have been uniformly sampled. This is true even of Slater's Virtual Light Fields [204]. Non-uniform light fields have typically been sampled with a smaller number of moving cameras, using techniques such as structure-from-motion to compute the parameters of the samples [27, 113]. This approach reduces the expense of light field capture in terms of apparatus cost and time. Non-uniform sampling however can also reduce the light field size, as sample densities can be increased or decreased depending on the complexity of the scene. Purely diffuse surfaces for example require far fewer samples than glossy surfaces. Khanna et al demonstrated some of their highest quality renders when augmenting a traditional rasterizer with a light field [104].

A non-uniform light field would not necessarily require novel parameterisations, but the bandwidth requirements would be considerable as each rendering ray would require a searching algorithm to identify appropriate samples. The large-word architecture used by current generation DFEs does not lend itself well to this type of algorithm. However, non-uniform sampling may be possible if an acceleration structure could be found that took advantage of the true-parallel access of the FPGAs local FMem.

3.5.3.3 Depth Correct Rendering

Double image artefacts due to depth discrepancies are the most significant artefacts of a light field renderer. This makes depth correct rendering an important feature [118]. There are depth correct rendering techniques using only highly localised data, making them ideal for implementation in a dataflow graph. Todt et al's approaches are not truly deterministic as they all involve some form of iterative search [227]. The loops could be unrolled into hardware, with the length placing an upper limit on rendering quality. The technique would still work even if the search were not allowed to run to its optimal conclusion. The issue with these searches is memory bandwidth. The depth value of a light field sample is required to estimate the rendering ray depth at each iteration, so no acceleration structure can substitute for accessing the light field itself every time. The number of accesses on a MAX4 DFE is at most 3-4 for each pixel in a pipeline driving a typical HD monitor, where the pixel rate is between 100-150 MHz. It is worth considering what modifications could be made to this approach to keep the design within one DFE.

3.6 Prototype 2D Renderer

The advantages of dataflow computing over traditional GPU implementations, is in their throughput and determinism. The algorithms towards the image-based end of the spectrum are those best suited to dataflow implementation. While dataflow graphs do not have the overhead of CPUs, the latency of the graph itself will not be much lower than the time it would take to execute an equivalent algorithm on a CPU. By levering the deterministic nature of the graph however we can apply it in such a way to bypass the sources of latency in a traditional display system, that of frame buffering. The high throughput and determinism allows us to compute pixels just-in-time, transmitting them as required at line rate, also known as *racing the beam*. This allows the system to respond visibly to user input much faster than traditional frame-based systems, as the user does not have to wait for the subsequent scan-out to see the consequences of their actions. To prove the feasibility of this design, we created a simple 2D image generator that computed pixels 'just-in-time' and drove a traditional LCD display without buffering a complete frame.

This is not the first dataflow renderer implementation of course. Ten Hagen et al. [222] presented one of the first practical examples of an image generator utilising dataflow computing in their Dataflow Graphics Workstation. The dataflow co-processor in their system did not drive a display directly, but performed pre-processing on 3D data, a function not dissimilar to modern day vertex or geometry shaders. Voitsechov & Etsion [236] present an alternative architecture for GPGPUs based on dataflow computing. In their architecture instructions from CUDA kernels are mapped to a dataflow graph. Their architecture supports multiple concurrent threads, using input token buffers at each node, which dynamically select which tokens to execute at any time based on available input parameters from the various tokens. This allows out-of-order execution at each node, maximising usage of the entire graph when some nodes (e.g. memory access nodes) have non-deterministic latencies.

FPGAs have also been applied to the specific problem of low-latency image synthesis. As seen in Section 2.5.1, Regan et al. [181] constructed a very low-latency 3D light field renderer. The architecture of their renderer was similar to ours. They achieved a latency of only 200μs, though the memory limitations of their platform permitted parallax in only one dimension. The closest example to our 2D prototype is that of Ng et al [155], who built a low latency direct touch interaction interface (described in Section 2.5.5).

Another popular application of dataflow computing in graphics is data visualisation. These applications take advantage of another benefit of dataflow computing - the ease and flexibility with which it allows the expression of complex series of data transformations, typically in a platform agnostic way. Dataflow computing is an ideal abstraction of the process of visualising large and heterogeneous datasets [215, 192]. Optimising flexibility and simplicity however are not priorities for the rendering applications we consider. The rendering implementations we consider are highly application specific and can be much more narrow in the type of data they operate on, and must meet far more stringent performance requirements. Our dataflow graph for example makes assumptions about characteristics such as the number of primitives, texture resolutions and colour representations. If these characteristics change, the graph is rebuilt and hardware re-programmed. By specifying these in advance however, we can minimise logic utilisation and latency.

3.6.1 Architecture

We constructed our renderer using a Maxeler Dataflow Engine (DFE). This is a computing platform for executing dataflow graphs described in Maxeler's high level language MaxJ. The architecture of our renderer is shown in Figure 3.7. The renderer computes pixel values by combining a set of sprites (2D images) of different sizes and locations. The DFE has two types of memory, 48GB of DRAM (off-chip) and ~5MB of SRAM (on-chip). The smaller sprites are stored in SRAM and the larger background maps in DRAM. The pixel colour values are computed by the Rendering

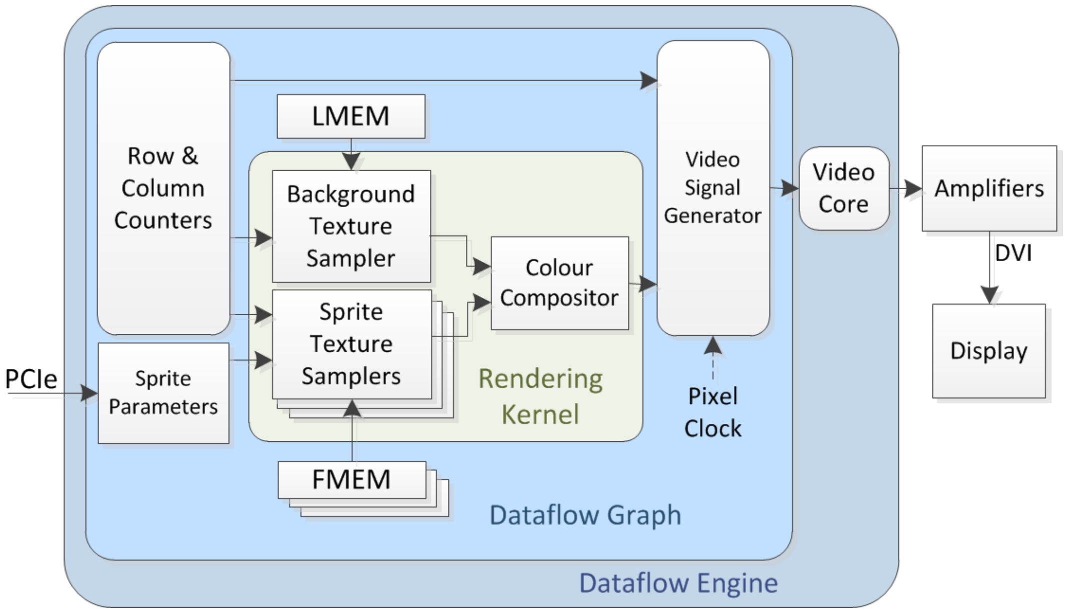


Figure 3.7: High level architecture of our dataflow renderer

Kernel. The current location on the display is tracked by row and column counters outside the kernel. For each pixel the Rendering Kernel samples colour values from the memories and combines them using a set of functional blocks operating in parallel. The final colour values are transferred to the Video Signal Generator, which generates timing signals such as HSync, VSync and DE. The combined colour and timing data is transferred to the video core, which performs 8b/10b encoding, and then line level encoding and output from the DFE using high-speed transceivers. The output is logically compliant DVI. A simple adapter board implements the physical interconnect. By synthesising the DVI signals and driving the display directly from the FPGA, we can ensure no additional latency is introduced and minimise the complexity of our apparatus.

Most functionality is implemented within the dataflow graph, in kernels running at ~200 MHz. The video signal generator runs at 152 MHz, the pixel clock rate of the 144 Hz display. MaxCompiler, Maxeler's toolchain, handles the transfer of data between kernels in different clock domains, as well as buffering and backpressure signalling between them. By controlling the size of the buffer between the Rendering Kernel and the Video Signal Generator Kernel, we can control how many pixels are rendered in advance. The buffer only has to be large enough to account for pauses in the data stream due to non-deterministic operations such as DRAM memory accesses. Pixel values are computed continuously. The only communication from the CPU is to update the algorithm parameters asynchronously. The parameters include which background map to use, and the location and content of the sprites.

3.6.2 Image Composition

The background map is sampled using burst reads into multiple DRAM modules simultaneously, with the reads being concatenated into a single 3072 bit word. Each word contains pixels within a segment of a line. The aspect ratio of these words are changed to form a stream of pixels. At the start of each new line in the frame, commands to read the next line in full are issued to memory. At the same time individual pixels from the previous reads are read from the input buffer. The commands to sample the first line of a frame are sent during the synchronization period, when nothing is drawn. For line widths that are not multiples of the memory width, the remaining data is discarded during the synchronization period.

Sprites representing buttons and cursors were so small they could be stored in SRAM, which supports fast random access. The address to sample is computed based on the offset into the sprite of the current pixel being computed. This is a function of the sprites location. The location and content of the sprites are updated via PCIe streams. If the offset is outside the bounds of the sprite, the sample is set to transparent. In most modules colours are represented as 32 bit RGBA allowing alpha blending. The sprites are composited in a fixed order obscuring or blending with the background map or those below them based on their alpha component. The alpha channel is discarded when the final colour samples are transmitted to the Video Signal Generator.

3.6.3 Display Interface

One of the biggest complications of the implementation was that the DFE was never designed to support a display interface. One option was to re-purpose an existing interconnect on the DFE such as Ethernet. The design of Ethernet however limits the available bandwidth, due to maximum frame sizes and header overhead. Further, a bespoke external device would be required to receive the Ethernet frames and drive the display. The DFE and this device would need to communicate to avoid buffer under or over-run. The DFE would need to respond very quickly to stall or resume requests, which implies a very shallow data graph. Otherwise, a large amount of buffering is required at the bespoke repeater, increasing the latency and possibly jitter. Given these complications, it was decided instead to re-purpose the physical layer of another interconnect and drive display data directly. The resulting infrastructure would only be of use to the specific model of DFE but it would allow development to begin sooner.

We re-purposed the MaxRing connectors on the DFE. These were connected directly to a set of high-speed transceivers on the FPGA. These transceivers incorporate generic Serialisation-Deserialisation (SERDES) [11] functionality, and can drive, within certain limits, any serial data. We re-purposed them to create a DVI display controller. Our display controller utilised eight high-speed transceivers present on the Stratix V powering a Coria DFE to transmit dual-channel DVI compliant data. DVI is a simple uncompressed digital display specification, which transmits 24 bit RGB colours across three synchronized physical channels. The DVI specification does not specify upper and lower limits for display data rates, but rather specifies that the signal must comply with the VESA timing standards [47, 234]. To give an example however, 60 Hz VGA has a data rate of 750 Mbit/s while WUXGA has a rate of 4620 Mbit/s. The Stratix V transceivers support differential signals at frequencies in the low GHz range. Signal amplifiers with output stages which were compliant with the DVI electrical specification were placed between these transceiver outputs and the DVI receiver. Custom logic was designed which received pixel values from a dataflow graph, formatted them into DVI compliant logical data, and transmitted them via the transceivers. MaxCompiler was modified to insert this logic into the design, and route the display data from the dataflow graph, through this, to the transceivers. We re-verified the end-to-end latency using the same technique as in Section 3.2.1.2, and confirmed the end-to-end latency of the system including our display driving infrastructure was under 1 ms.

3.6.3.1 Logical Display Interface

The colour and timing data is encoded into logically compliant DVI in the Video Core. DVI is a fully digital protocol which transmits 24 bit RGB data over three serial channels, with a fourth transmitting the pixel clock. The 8 bit colour words are converted to 10 bit words with a specialised version of 8b/10b encoding, ensuring each channel remains DC balanced [47]. During the blanking periods of the frame, the RGB data is substituted for control data, which includes the HSync and VSync signals.

The Data Enable (DE) signal from the Video Signal Generator determines whether the colour or control data (Hsync & Vsync - also from the Video Signal Generator) should be transmitted on a given clock cycle. The colour or control words are routed to an 8b/10b encoder and then into the serialiser of the high-speed transceivers on the FPGA of the DFE. The words pass through a shallow FIFO buffer used to transfer the words between the clock domains of the Video Signal Generator and the transceiver Physical Media Access (PMA), which run at the same rate but may be out of phase.

Maxeler's toolchain does not currently support direct access to the transceivers, however with Maxeler's assistance a small modification was made to the toolchain to give access to our dataflow graph. The Physical Coding Sublayers (PCSs) within the transceiver blocks were configured to provide direct PMA access to the transceivers. The serialiser is present within the PMA of the transceivers, and was configured with a serialisation factor of 10. The transceivers were bonded together and driven by a single external serial clock, produced by a fractional-PLL placed within the transceiver bank. A fourth transceiver was used to transmit the pixel clock. To do this a constant pattern of 0x1Fh was written to the parallel data port. The driver output stage is shown in Figure 3.8.

3.6.3.2 Physical Display Interface

The transceivers on the DFE use Pseudo Current Mode Logic (PCML) and are AC coupled. DVI specifies DC coupled CML, with the common-mode voltage set by the receiver. To make the output DVI compliant, a board was constructed which

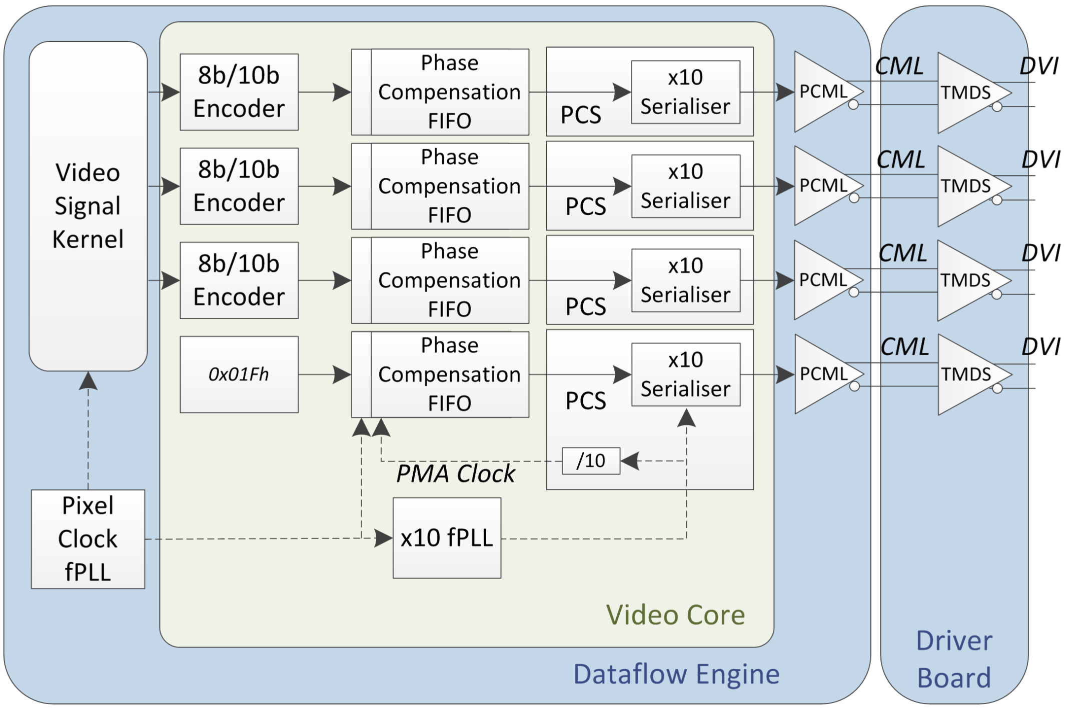


Figure 3.8: Diagram of the data transfer and clocking of our DVI driver output stage, and at what level each component exists

routed the serial data through a TI DS34RT5110. This is an HDMI re-timer IC which implements Transition Minimized Differential Signalling (TMDS) [100] outputs, the signalling technology specified by DVI. The board attaches to proprietary gold finger connectors on the DFE, and presents a standard DVI female connector. Its design and layout was based on the reference provided by Texas Instruments [224]. An image of a second-generation apparatus is shown in Figure 3.9.

3.6.4 Latency

Figure 3.10 shows our prototype renderer driving a typical LCD display with a solid column. The column is being moved to the left and right by the user with the mouse. As can be seen the 'frame' is being updated during scan-out as the latest position of the sprite is being updated on the DFE repeatedly by the CPU. The approximate sampling rate of the mouse can be determined by the number of lines scanned out between sprite updates. A ghost of the previous frame (when the mouse was static) can still be seen due to the high persistence of the LCD panel.

We measured the end-to-end latency of our apparatus at ~6 ms, using the cross-

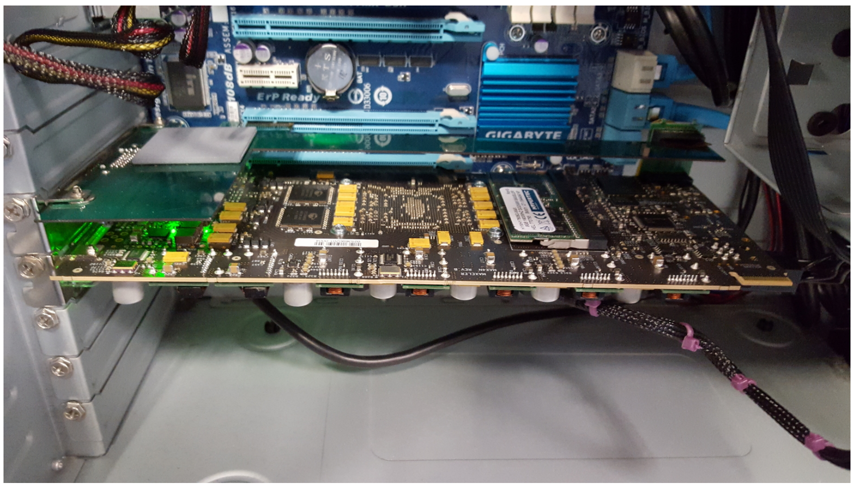


Figure 3.9: Maxeler Isca DFE with DVI interface board attached above

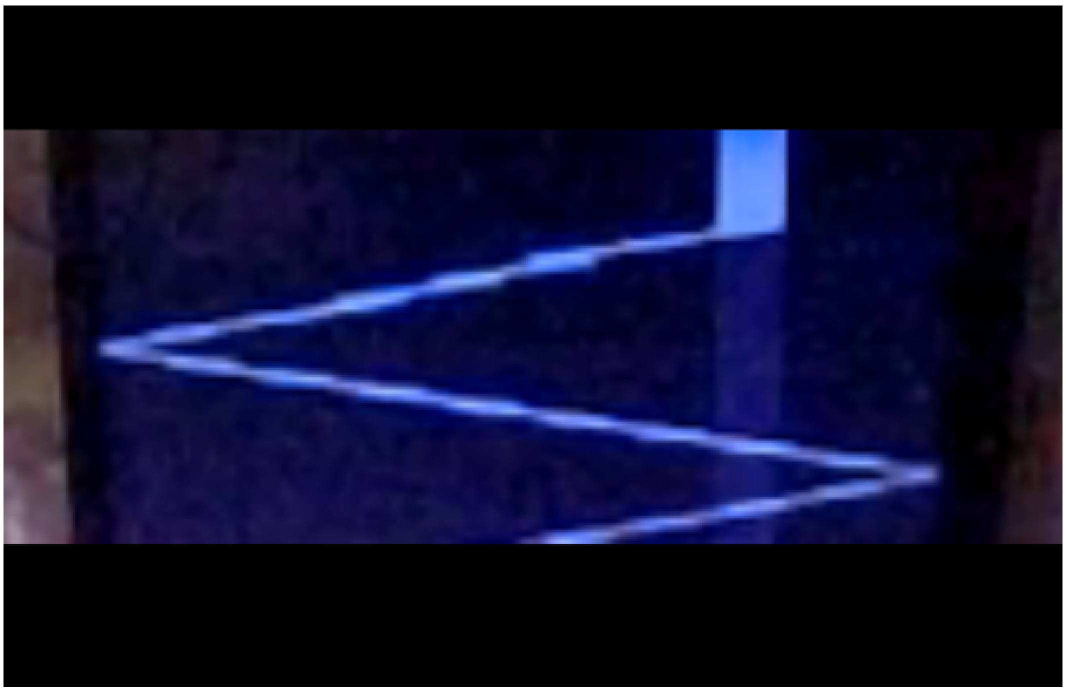


Figure 3.10: Photograph of our 2D prototype renderer driving a sprite moving back and forth on a typical LCD

correlation variant of Steed's Method [62]. This was predominantly the scan-out time of the display. In addition, we connected LEDs to a specially designed output of the DFE, and the parallel port of the host PC. On receipt of a specific input the CPU application would signal the parallel port and the DFE to illuminate these. High speed video (1000 fps) from a Casio Exilim ZR1000 digital camera monitored the input devices and the LEDs. High speed video was used rather than an oscilloscope because it could monitor the input device and scan-out in addition to the LEDs, without additional instrumentation. The delay between each stage was determined by counting the number of frames between the occurrence of each event. The high speed video showed the first LED activating within ~1 ms of the input device being triggered. The delay between the first and second LED was so low it could not be discerned from the video. This is summarised in Figure 3.11.

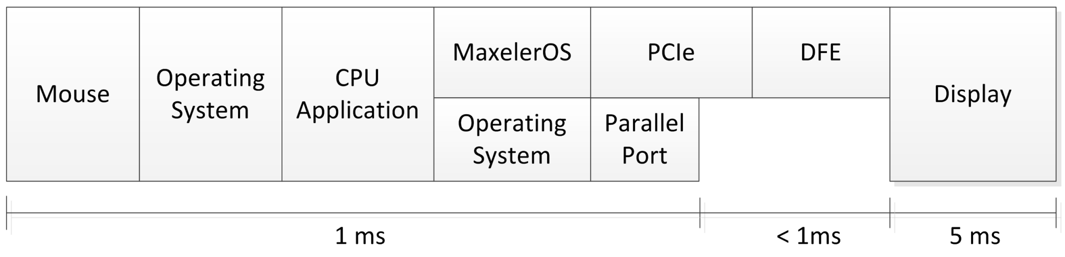


Figure 3.11: Total observed delay of each stage of our apparatus

3.6.4.1 Comparison with GPU

We rebuilt our apparatus with a GPU (an NVidia Quadro NVS 290) in place of the dataflow renderer in order to measure the latency a traditional GPU would provide. The system remained otherwise unchanged. A small program was written which drew three textured squares, one controlled by the mouse. The program used the GLUT toolkit to draw quad primitives specified directly in normalised device coordinates. The latency was measured under two conditions using the cross-correlation variant of Steed's Method [62], that uses the relative sub-sampled motion of two tracked points to estimate the latency between them. The results are shown in Table 3.1. We used a swap chain of length 2. With no swap chain the tearing artefacts were so severe measurements could not be taken.

Table 3.1: Average and Standard Deviation of the latency of our stimuli when rendered with a GPU

Condition	Average (ms)	Std. Dev. (ms)	Repeats
VSync On	26.17	2.79	6
VSync Off	19.86	4.16	7

3.6.5 Summary

Reconfigurable hardware has been used previously for low-latency image synthesis. These are typically low level implementations with tight vertical integration. For example the apparatus of both Regan et al. [181] and Ng et al. [155] had the tracker driven by the same device performing the rendering. Reconfigurable hardware combined with the dataflow programming model can make application specific rendering hardware cost effective. Our sprite renderer has comparable scope to both prior examples, but our dataflow graph can be adapted to other use cases with an effort comparable to GPU shader programming.

Dataflow computers could be an ideal platform to create new renderers without the limits of the painter's algorithm. Our renderer avoided the buffering inherent in that algorithm, and ran asynchronously of the CPU. Its architecture allowed us to race the beam, minimising the delay between user input and what is being drawn to the screen. For the renderer itself, this was less than 1 ms.

Our implementation currently requires a modified toolchain to drive the display. An alternative would be to use a standard platform interface such as Ethernet to route display data to another device. So long as the chosen protocol included backpressure functionality the design would remain conceptually the same, but with the Video Core and display driver implemented externally. The design would be more complex however, with far more buffering and therefore higher latency.

In the next section we discuss how this architecture was extended to support frameless rendering of 3D primitives in order to create immersive virtual worlds.

Chapter 4

Real-time Ray Caster

In this section we describe the implementation of an ultra-low latency 3D renderer using the same just-in-time technique as employed in Section 3.6. The purpose of this prototype was to lay the foundations for a purely image-based renderer. Out of the approaches to rendering considered, this the technique that maps best to the dataflow concept when just-in-time rendering is used. As described in Section 2.4, image-based rendering is not inherently deterministic, but implementations are typically more so than other techniques. This is because the sampling operations are more constrained and are localised to smaller areas of the screen. Combined with an implementation in which the sampling is deterministic, such techniques will bypass one of the largest sources of latency - the scanout synchronisation as described in Section 5.3.2.

4.1 Frame-less Rendering in Virtual Reality

Latency cannot be removed given current technology, so a number of authors have designed methods to compensate for it. We have already reviewed Regan & Pose's Address Recalculation Pipeline in Section 2.5.2. Their implementation allowed rendering from environment maps at very high rate, creating the illusion of low latency for orientations. This is not dissimilar from the architecture proposed by Lincoln et al. [119] and Zheng et al. [263] - that of a cascade of increasingly simple but fast image warps. Regan & Pose's implementation however is one of a cascade of renderers, moving towards the image-based end of the continuum as they go. Other authors have taken this approach as well.

Mark et al. [133] created an architecture performing 3D post-rendering warping. In this system, a traditional renderer created a set of reference frames containing depth information. One of two possible reconstruction techniques (planar-to-planar warps defining per pixel disparities, or a deformed mesh imposter) could then be used to composit these reference frames into a new image. The reconstruction stage is computationally far simpler than rendering the entire scene, so it allows the system to appear to respond more quickly. Mark et al.'s system through the use of depth information supports changes in both orientation and translation. An exact latency was not reported but Mark et al. state it would be only the time required to perform a 3D warp. Smit et al. [212] pursued this image warping architecture. Their system created a mesh - a grid of vertices with a count equivalent to the reference image resolution. A typical GPU was then used to deform this mesh with a vertex shader implementation of an image warping algorithm. The fragment parameters of the resulting frame could then be used to sample the original image. The image could be reconstructed in a number of ways, however Smit et al. found that treating each vertex as a point and performing screen space point-splatting, with the point size dependent on the depth, provided the best trade-off of speed and quality. They measured the latency of a single-GPU implementation (rendering and then warping on the same device) as 15.7-17.1 ms, depending on scene complexity. The latency of a multi-GPU implementation (one rendering, one warping) was higher at 50.8-57.4 ms, but much less sensitive (lower standard deviation) to scene complexity.

Li et al. performed full depth image warping on an FPGA [116]. They did not report a latency but determined their circuit could run at up to 88.152 MHz. Yanagida et al. [256] proposed using a rate gyro to determine the corrections required for a latent image from the point at which rendering began to right before it was displayed to the user. The image underwent a simple shift and rotation to account for the change in orientation detected by the gyro (similar to Lincoln et al.). The authors used a gyro as it was faster than the absolute magnetic trackers used to render the reference frame. The authors simulated the effect of their system on image quality but did not build the entire system so the latency is unavailable. Our design is a ray-caster, and it does make use of techniques such as mip-mapping and caching to exploit ray-coherence, however it has more in common with the designs of Li et al. and Regan & Pose, than hardware-accelerated ray-casters. This is because our design performs no shading and limits the traversal depth to minimise latency, making it more like a lumigraph renderer than a ray-tracer.

One issue with compensating for latency with image warping is that typical techniques use a short history of one or two reference frames with which to compute new images. If the user moves too quickly, or the rate of reference image generation is too low, missing information will result in holes in the warped image. The concept of the golden thread introduced by Bergman et al. [19] (Section 2.4.4) is compelling due to its mapping to parallel hardware. Their implementation however approximated this with multiple rendering modes that operated on a frame-by-frame basis, which is not appropriate here. Qu et al. [179] had an interesting interpretation of this concept. They combined 3D warping and ray-tracing in their voxel renderer. The authors' cascaded system performed a 3D warp of a keyframe, and would then fill in any missing information by ray-tracing those specific pixels. The latency was not reported, as the authors' test implementation was designed evaluate quality rather then speed.

4.2 Design of a real-time Ray Caster

We chose to implement a real-time ray caster that would draw an environment map. The scene geometry would be defined ahead of time and the detail would come from the texture maps, as in Regan & Pose's implementation. We chose this implementation as the development time for a more advanced renderer, such as a light field renderer, would be considerably longer. The concepts are well defined and researched, but there are few working examples. Further, environment maps were good enough to support the immediate experimental use cases. It was expected that the working ray-caster could be extended in one of two ways. It could be refined to be entirely image-based, creating a light field renderer. Alternatively, it could be augmented with the ability to receive streamed environment maps from a GPU, perhaps with depth information, turning it into a 3D warper stage.

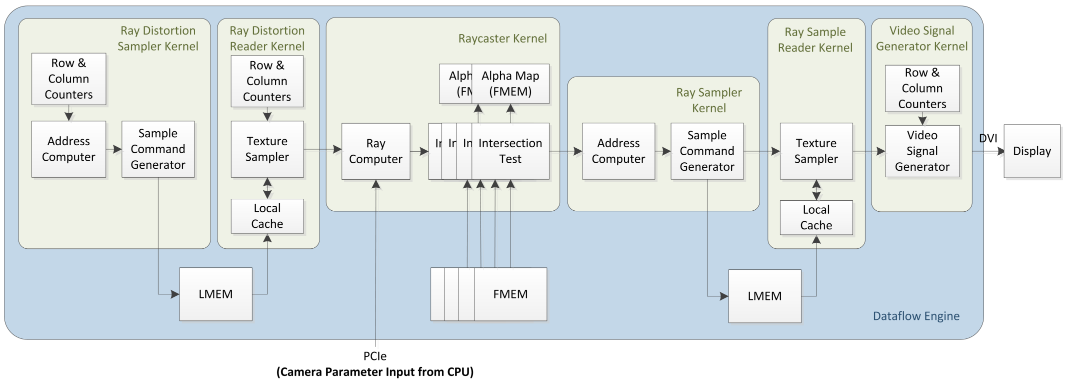


Figure 4.1: Diagram of the ray-caster renderer architecture

We implemented our 3D renderer on a Maxeler ISCA DFE. Our algorithm begins with a set of counters which identify the location on the physical display to be rendered. The Raycaster kernel receives the viewport locations, and computes the parameters of sampling rays in the traditional way using the camera properties sent asynchronously over PCIe. It then performs intersection tests between these rays and a set of planes. Each intersection test is a separate series of operations in hardware and the plane parameters are defined at design time. The result of each intersection test is compared with that of the one before it, and the result of the closest intersection is propagated to the next test. Once the closest plane has been identified, the intersection point on its surface, and then a set of UV coordinates are computed. The UV coordinates are converted into a memory address by the Ray Sampler Kernel. This address is sampled by the Ray Sample Reader Kernel. The resulting colour value is combined with timing signals in the Video Signal Generator Kernel and transmitted via DVI to the HMD. A diagram of the dataflow graph is shown in Figure 4.1.

In newer COTS HMDs such as the Oculus Rift DK2 (the one used in this prototype apparatus), the virtual world is rendered with a wide field-of-view (FOV) and then distorted - compressing it so that it will fit on a typical form-factor display. Lenses in the headset undo this compression making the scene again appear to wrap around the users entire FOV. This distortion is typically done in a post-processing stage on a typical GPU. As the distortion consists of a per-pixel mapping between a location on the real display and a location on the virtual viewport however, it is trivial to incorporate equivalent functionality into our design. The Raycaster Kernel takes in arbitrary positions on the virtual viewport, and outputs a stream of pixels. Therefore we can define a map, which maps real screen locations (identified by the counters) to distorted locations on the virtual viewport, and feed these distorted locations into the Raycaster, receiving distorted samples at the real locations at the output. The Ray Distortion Sampler Kernel and the Ray Distortion Reader Kernel are responsible for reading the per-pixel disparities from a distortion map, and feeding the locations on the virtual viewport onto the Raycaster Kernel.

Both the distortion map and the environment map are stored in DRAM (LMEM). On a DFE multiple DRAM modules are concatenated to form one very wide (1536 bit) address space (LMEM). DRAM is low cost but has high latency and so caching is used to maximise bandwidth utilisation. Both the distortion map and environment map are split into tiles and these are read from DRAM using burst accesses. When LMEM is accessed, upstream kernels generate read commands and downstream kernels read the resulting data. The addressing logic is duplicated in both, so downstream kernels can predict what commands the upstream kernels will send and therefore whether their current cache is valid, and what tiles they can expect to receive subsequently. In a dataflow graph all tokens are executed and transmitted in order. Kernels only run when tokens are available at all inputs and space is available at the output. By duplicating the addressing logic, the upstream and downstream kernels do not have to be explicitly synchronized around the non-deterministic accesses into LMEM. Similarly, because pixels are generated in the same order as they are scanned out to the display, the Video Signal Generator kernel can maintain the current location on the physical display using its own counters, without any direct connection to the first kernel.

The distance across the surface of a plane between two subsequent ray intersection points depends on parameters such as field of view, and distance between the camera and the plane. These cannot be assumed ahead of time and so mip-mapping is used to ensure that for each memory read, on average, 8 subsequent samples can be read from cache. Due to the high latency of DRAM if this were not done the memory would not be able to keep up with the sample requests and the display would be starved of data. The mip level is recomputed for each pixel based on the distance of the current intersection point on the plane, from the previous intersection point.

If we were to extend this design to be a 3D warper, it would be best to move to faster off-chip memory such as QDR, or even SRAM. This would be much more expensive than DRAM, but the exact memory requirements for an environment map would be known ahead of time. These faster memory technologies would somewhat relieve the bandwidth constraints, allowing enough to stream in and out as the map is updated. For light fields, a very large amount of memory[1] is required, and so DRAM with an effective caching scheme would be most suitable.

The design uses a cascade of clocks which decrease in frequency further down the graph. This ensures that data such as memory read or write commands are always produced faster than they are consumed. This serves two purposes (1) the buffers are always full ready to smooth out interruptions in the data and (2) when a section of the graph is required to stop or resume, the time it is down for is minimised (as the higher clock speeds propagate the stall signal faster) reducing the loss of bandwidth due to stalls. On FPGAs, there are a smaller number of logic element configurations that will work correctly, the higher the speed of the clock driving them, making it ‘harder’ for the toolchain to fit the design into the device. It is convenient in the case of our algorithm that the upstream functions, such as the display counters and distortion map sampler, are simpler and therefore easier to fit. In the ray caster kernel, serialising the primitive intersection tests (rather than laying them out in parallel and accumulating the results in one stage) also reduces the distance a given signal will have to traverse unbuffered and therefore improves fitting time. The final stage in the graph, the data sink, is the DVI transmitter, which runs at the display standard line rate (165 MHz for the Oculus DK2 display). Most other functions are run at 200-210 MHz.

[1]Tens to hundreds of Gigabytes, for a small room, following the guidelines of Gortler et al. [67]

4.3 Virtual Worlds

Currently, the design uses a series of planes as the primitives. As each plane test has its own hardware intersection test resources it can be augmented with different functionality. For example some planes support visibility maps. These are stored as low resolution 1 bit maps in SRAM (on-chip) and can mask a plane's visibility based on its UV coordinates. Alternatively planes can respond differently if a ray is cast from the front or back. The primitives do not have to be planes either. Any simple primitive, such as a sphere, can be easily represented, so long as the ray-intersection test can be described in a deterministic set of operations. The primitives can also be updated via the CPU in the same way as the camera parameters allowing them to move. These sort of features can be used to create more dynamic VEs (for example the Pit Room shown in Figure 4.3. The renderer is not intended for arbitrary scenes however. Ray-intersection tests are expensive and logic is limited on the FPGA. The best use case is to keep the detail in the maps, which are more easily updated.

We implemented a basic cubic environment map to begin with using six planes (see Figure 4.2). A spherical environment map may appear to be simpler, but implementing the transcedental functions in hardware is non-trivial. The typical way to do this is with look up tables, that would have to be considerable in size to support high resolution maps. We also implemented a pit room environment (see Figure 4.3). Recall that the ray-caster is not a ray-caster in the traditional sense and does no lighting calculations. Therefore any light modelling should be baked into the maps, such as shown in Figure 4.4 for the pit room environment.

4.4 CPU Application

One of the advantages of the renderer is that unlike GPUs it runs asynchronously to the CPU. As the scene composition is complete and persistent, the CPU only needs to communicate when this or the viewing parameters change. Still the CPU application communicates with the renderer at a very low level. For example, when the camera parameters are updated the CPU will transmit an update command consisting of a set of basis vectors, rather than a position & direction. The CPU application

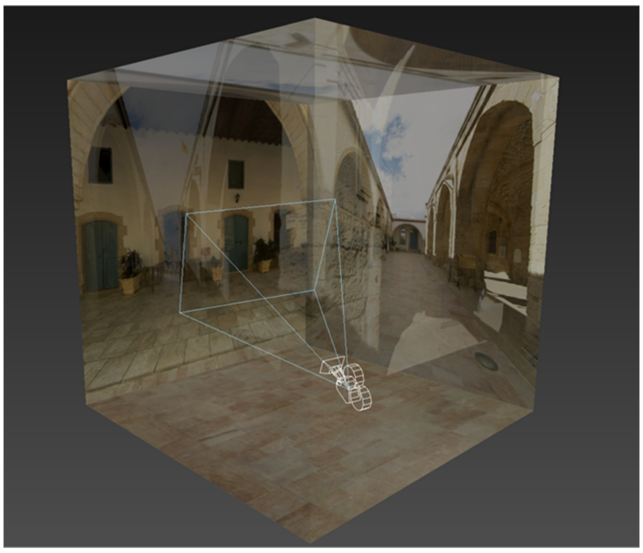


Figure 4.2: Scene geometry proxy used in the environment map real-time ray caster renderer

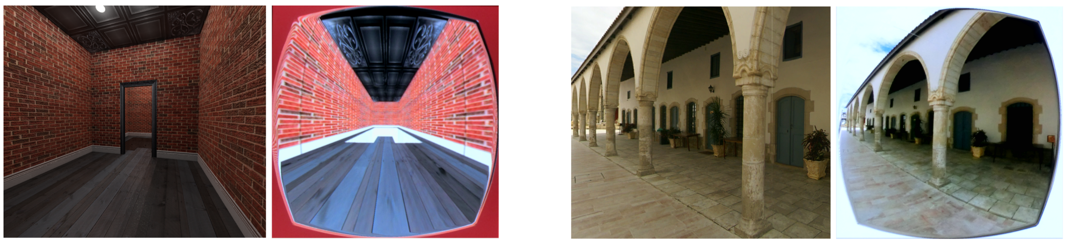
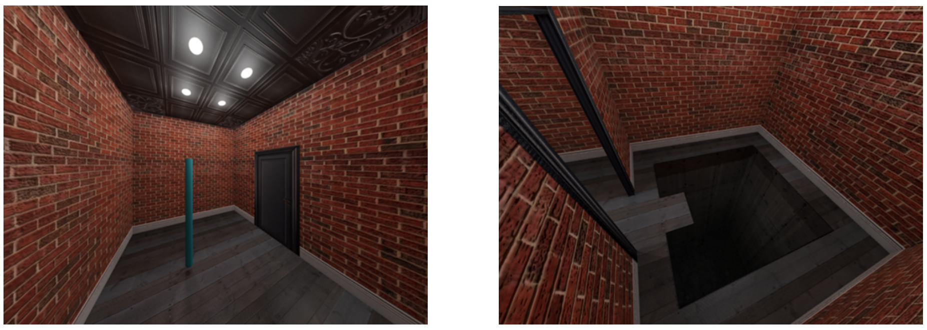


Figure 4.3: Images of two virtual environments designed for the DFE frameless renderer, along with photos of the right eye region of the display showing them being drawn in real-time. The Pit Room (left) is a synthetic environment made up of nine alpha-mapped planes. When it is complete, global illumination and shadows will be baked into the textures, as shown in the leftmost image. The Lazarus environment (right) is a six sided cube map captured from the real world, and is the one used in the current experiment.



(a) **(b)**

Figure 4.4: Pit-room environment with baked texture maps

has an abstraction layer approximating a primitive graphics API, which transcodes geometry, textures and rendering parameters into structures suitable for writing directly to the renderer. Communication is via a set of low latency PCIe streams. That is, the CPU application transmits words over PCIe as required. These streams connect to specific locations in the dataflow graph, where their parameters replace registered values used to compute the pixels.

The PCIe channels are configured to use polling to reduce the latency as far as possible. There is little to no buffering within the renderer itself and so almost all the latency is in the CPU application. As the CPU application needs only to be concerned with updating the tracking data as fast as possible, one real-time priority thread is created for sampling to the tracking system (the mouse, or the Oculus DK2 Inertial Measurement Unit (IMU)) and updating the renderer. The loop in this thread is carefully written to avoid any blocking calls, or those that result in yielding (e.g. terminal IOs). The tracking systems are polled, like the PCIe interface, to avoid OS event handling overhead. This effectively gives control of one core of the multicore processor in the PC over to the thread. This can be verified by repeatedly polling the systems high performance timer and checking the period between queries.

4.5 Summary

In this section we described local extension of the 2D renderer described in Section 3.6 to 3D. Ideally we would build an image-based renderer, in which any geometric information is implicitly defined. This is because the sampling can be performed with a smaller number of more highly predictable operations. Reducing uncertainty in how the algorithm will execute increases determinism and predictability of performance. This leads to a more efficient use of FPGA resources. It also allows techniques such as racing-the-beam which bypass one of the largest sources of latency in current graphics systems.

The ray-intersection test used here is an implicit surface definition, and does begin to approximate an ideal image-based renderer to an extent. The way they are combined however is not ideal. This is because the plane itself is not an appropriate primitive to use to represent generalise worlds. For the time being however, this design allows the system to meet its other objectives. In Chapter 5 we describe an evaluation of this system, from the perspectives of latency and quality.

Chapter 5

Evaluation of Low Latency Renderer

In Chapter 4 we described the design of a real-time ray-caster. In Chapter 5 we describe the evaluation of this renderer in terms of latency and visual fidelity. The purpose was to understand quantitatively the effects on what the user perceives. The motivation was both to gain understanding of this, since it is not a well-studied area, but also so we could better understand any effects observed during our immersive user study described in Chapter 7. The latency was measured using high speed cameras, while the fidelity was measured using algorithms known as image quality measures.

5.1 Quantifying Visual Quality

When building systems that trade-off spatial quality for speed, authors need a way to quantify the performance of their renderer. As discussed by Ferwerda [54], there are a number of ways to define quality or realism when discussing synthetic images and virtual reality. VR attempts to substitute virtual stimuli for real, and therefore one metric would be sensory believability [105]. Current technology does not have the ability to re-create the full range of visual stimuli that would be required if we were to compare a render to the 'real world' however. Most authors of frameless rendering techniques compare the performance of their renderers to ground truth renders - images produced off-line, simulating an ideal renderer with zero latency. Various metrics are used to describe the fidelity of the first image to the latter, for example Mean-Squared Error (MSE) for static images (e.g. [241]) or Fast Fourier

Transforms in the spatial domain for dynamic images (e.g. [260]).

As illustrated by McNamara [141], objective Image Quality Measures (IQMs), like MSE, can deviate significantly from what a user would consider to be a correct characterisation. Accordingly, a number of authors have proposed measures which are based on the operation of the human visual system. These are intended to give more weight to salient artefacts while minimising the influence of less significant ones. Wajid et al. [237] compared a number of these measures to determine which could most accurately predict the Image Quality Assessments (IQAs) performed by human participants. They found that MSE & Peak Signal to Noise Ratio (PSNR) were acceptable but Visual Information Fidelity (VIF) ([196]) performed best. However, by considering only those artefacts which are subjectively most significant, we risk missing an unexpected interaction or misrepresenting the fidelity of the image.

For example, motion blur is a rendering artefact that reduces image quality when compared with a ground truth render, but it is also a good visual cue that improves user experience when it stops animation being seen as jerky or strobing [153]. Čadík et al. [31] performed a similar study to Wajid et al. but focusing on the performance of Image Quality Measures (IQMs) in assessing artefacts common in synthetic CG imagery, noting that traditional IQMs are often tuned for compression and transmission artefacts. They confirm that no single metric performs steadily for all tested stimuli, though sCorrel (Spearman's Rank Correlation Coefficient over 8x8 pixel blocks) performs relatively well. Zhang & Wandell [262] performed a similar study comparing the simpler Root Mean Squared (RMS), CIELAB and their own spatial-CIELAB ([261]) metrics. They found that the sCIELAB predictions were significantly better than RMS. It is important to consider that various IQMs will have inbuilt biases, and to select a range of measures.

5.2 Apparatus

In order to evaluate the renderer, it had to be integrated into an apparatus approximating a virtual environment. The renderer was connected to an HMD and software was written to display a virtual world. The system was instrumented so that its internal state and the display the user would see could be recorded. It also required a system to compare with.

5.2.1 GPU Renderer

In order to compare how the behaviour of our frameless renderer deviated from a typical VE, we constructed an equivalent system, but using a GPU in place of the DFE. Our GPU system consisted of a typical PC running Windows 7 with an NVidia GTX 680 GPU. The CPU application was a modified version of OculusRoomTiny, a reference design included in version 0.4.4 of the SDK for our HMD. We modified this application to remove the input processing stages and have it draw a cube made up of six planes surrounding the user's viewpoint. We also removed a feature which used the latest tracking data to adjust the mesh used to apply the lens distortion before post-rendering warping. This was because these types of warps may introduce unpredictable spatial distortions as they do not take into account the relative depth of the warped points in the scene. Further this functionality gives it some features of a 3D warping architecture, and our aim was to make a comparison with a typical GPU system. The use of techniques such as View Bypass and Time Warping which are supported by the SDK can be very effective at reducing apparent latency however. This is discussed in the conclusion.

5.2.2 HMD and Tracker

The HMD we used was an Oculus Rift DK2. For capturing head motion, we used only orientation data. It was captured from the on-board IMU, which connects via USB and updates at a rate of 1 kHz. To perform the head motion captures we used the same GPU system described in Section 5.2.1, however it was modified so that tracker was read in a separate thread not restricted by the GPU. The timestamp of each sample was considered to be the timestamp already assigned by the SDK when it was read in. The mean interval over the entire capture was 1.5 ms, though 22% of the samples had an interval of 1 ms or less. The tracker then is capable of running at 1 kHz, though some variance is introduced between the device and our code. As a result an 18 second sequence is covered by approximately 12100 samples.

The DK2 features a 1920×1080 portrait display orientated on its side connected via HDMI. HDMI is backwards compatible with DVI using a passive physical adapter. The screen is split so that each eye sees 960×1080. The screen is a low-persistence OLED display. The display has a typical sequential scan-out but the pixels are only illuminated for a short period, resulting in a 'rolling band' of illuminated pixels. This can be seen in Figure 5.3. Note that the width of the band in an image such as 5.3 depends on the exposure time of the camera as well, so the true width may be narrower. For our experiment, once the head motion had been captured we dismantled the HMD and secured the display to a camera rig (Figure 5.1). At this point tracker data was provided from the logs and only the display was used.

5.2.3 Synchronisation LED

To measure the latency of the rendering stages of our systems, and to synchronise the video captures of the HMD with the tracking data, we needed to be able to instrument the CPU code. To do this we used an Arduino Uno, a micro-controller development board, to toggle an LED on command via serial link over USB. To ensure that the latency of the serial link, and rise & fall times of the LED were trivial (~10*ns*), we configured the Arduino to loopback all commands, and the CPU code to block until receipt of these echos. The CPU then cycled the LED as fast as it could, while it was monitored with a 1000 fps camera. The total round trip time for two commands was ~3 ms on both Windows and Linux, much shorter than the frame period of rendering captures. The loopback was disabled during captures of the renderers running in real-time.

5.2.4 Cameras

To measure the latency and confirm correct operation of our apparatus we used a Casio EX-ZR1000 consumer digital camera. This camera is capable of capturing 224*x*64 video at 1000 fps with a rolling shutter. To capture the rendering systems in operation for image quality analysis we used a PixeLink DL-D722CU-T USB3 camera. This camera had a configurable exposure time down to 1 ms, a global shutter and could capture at up to 257.7 fps. The screen, camera rig and synchronisation

LED are shown in Figure 5.1.

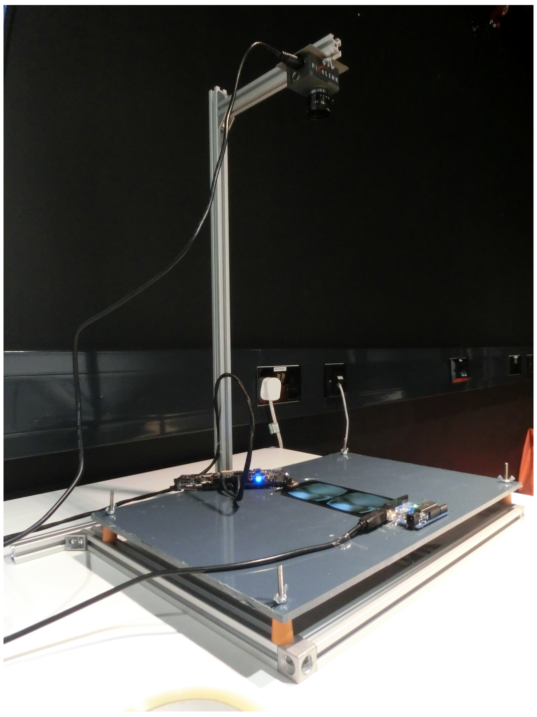


Figure 5.1: Image of the apparatus showing the low speed camera, Arduino and DK2 screen

5.3 Results

5.3.1 Latency

We measured the latency of the rendering stages of both GPU and DFE systems. To do this we configured the CPU application to cycle between two viewport orientations values, at a rate of 1 Hz. When changing the orientation, the CPU cycled the synchronisation LED to indicate exactly when it had updated the state of the rendering system. 1000 fps video was taken, with both the display and the LED in view. The latency was measured by counting the number of frames between the transition of the LED and the first change in the content of the display (the 'tracker to beam' latency, or, the latency of the rendering stage not including the scan-out time). The technique is similar to those described by He et al. [73] and Di Luca [44].

The GPU system had a latency of 25.7±0.4 ms. This is not unexpected. The display refresh rate is 75 Hz and the CPU application we based our system on syncs its main loop to this. One frame period is 13.2 ms. It will take the CPU one cycle to issue the commands to render, and once complete the GPU will wait for VSync before the frame is swapped to the display. In addition no less than 13.2 ms of latency will be added as the scan moves across the screen. This is because the GPU system scans out a single frame at a time.

The DFE system had a latency typically lower than the the temporal resolution of the video. At this level the synchronisation LED latency becomes non-trivial, so we cannot say the latency of the rendering stage is less than 1 ms. It takes the frameless renderer less time to read a tracker value and update its state than it does to scan out, therefore the latency will be 1 ms at the location of the scan-beam, and 13.2 ms at the location about be overwritten by it.

5.3.2 Rendering

To better illustrate the differences resulting from the alternate rendering techniques, we applied a simple grid as the texture of our six-sided environment. The CPU was then configured to pitch the camera up and down through 180°at 2-17 Hz, while the display was captured. The DFE system continually updates the tracking data every few lines. The latency is therefore lowest at the point the 'beam' is scanning across the display, and highest at the oldest visible pixel. The frame-based GPU system draws a static image produced for some previous tracking sample. The latency is therefore lowest at the point that scan-out begins (the shortest time between the production of the frame and it beginning to be visible), and increases as the beam moves across the display, as the frame being scanned is ageing while the scan-out proceeds.

If we look at a 13.2 ms exposure capture, we can see clearly the differences in the approaches. The DFE system results in a skewed image under high velocity, never displaying a whole frame for a single tracking sample. The GPU does not have these skewing features, but at the cost of a much greater time between, and therefore difference in, subsequent frames. This is shown in Figure 5.2. Figure 5.2 has been annotated showing the latency between regions of the display and the current tracking data (i.e. the ideal tracking data at the time the capture was taken).

C The current location of the scan-out. The latency of the DFE is ~1ms. At 144

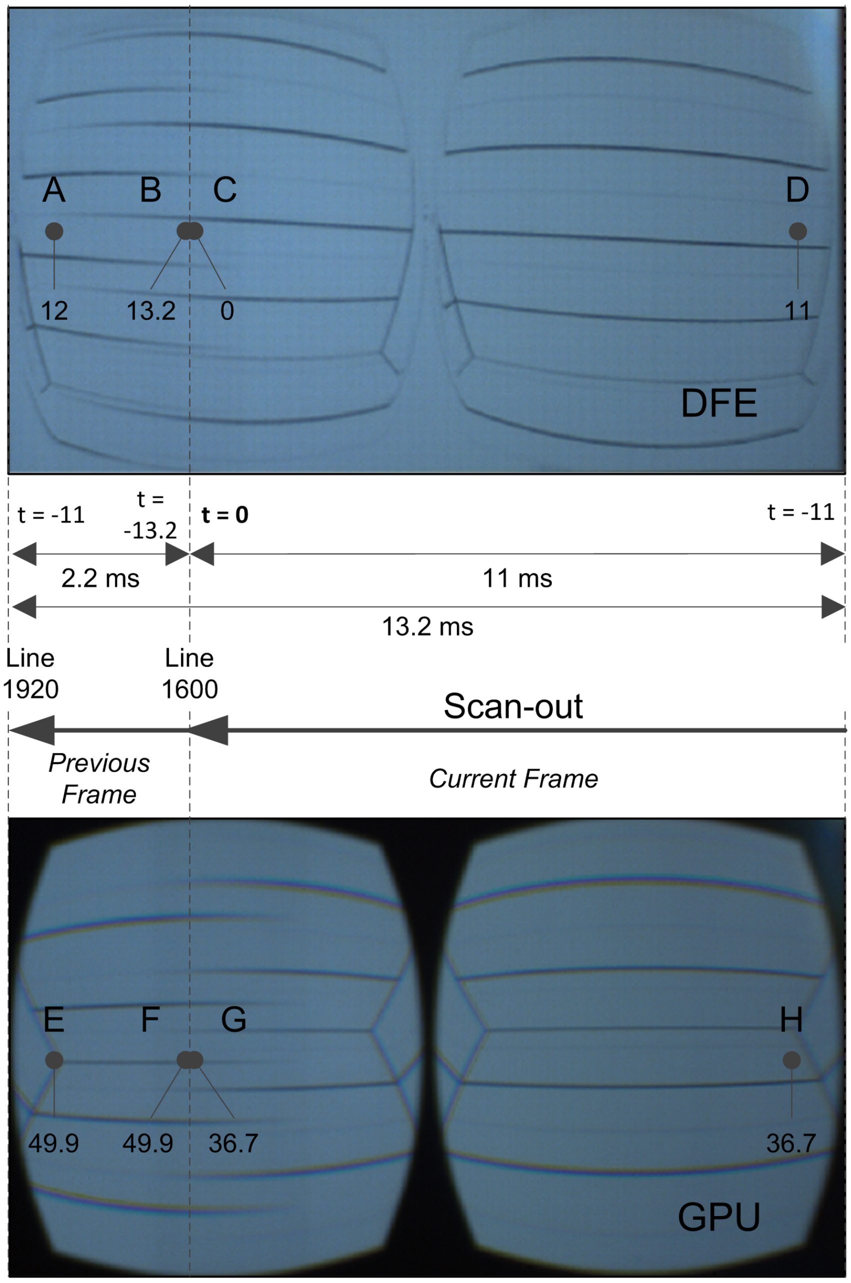


Figure 5.2: Captures of the DK2 screen with a 13.2 ms exposure, drawn by the DFE and GPU while in motion, annotated with the latencies at four different locations (A,B,C,D & E,F,G,H) on each frame for an arbitrary point in time during scan-out (at line 1600)

lines per ms the vertical region around C has the lowest latency of any on the display. This is because pixels drawn around region D, for example, were computed for older tracking samples than those at C.

D The start of the scan-out for the display. This has less meaning for a frameless renderer than a frame-based renderer since the sampling of the tracker is independent of where the scan-out begins.

B,A The scan loops round immediately from A to D, so region B is the oldest region on the frame, last drawn 13.2 ms before the latest tracking data.

H This is where a new frame from the GPU begins. The GPU has a latency of 25.7 ms, so when a scan-out begins the frame is already 25.7 ms old. If we were to capture when the scan was at H, the latency at H would be 25.7 ms.

G As the scan moves across the screen, the 25.7 ms old frame is ageing as it goes. By the time the scan reaches region G, the frame has aged an additional 11 ms. Any part of the display showing the content of that frame (between and including regions H and G) is therefore showing data 36.7 ms old (25.7 + 11), and this latter number will increase as the scan proceeds through F and H. On a V-Synced frame-based system, scan-out always begins at the same place, so these delays across the display will be the same for every frame.

E,F These show the previously rendered frame as they have not yet been overwritten. The previous frame continues to age while the new one is being scanned out, so the latency of any region showing this content is the latency required to complete and draw the previous frame (25.7 + 13.2) plus the time to reach the region in order to overwrite it with the new one (11): (25.7 + 13.2 + 11 = 49.9).

The DK2 uses an Organic Light-Emitting Diode (OLED) display. As an OLED display, the individual pixels have transition times much faster than those of LCDs, closer to that of CRTs. The low persistence of the display is facilitated by rolling scans, in which the screen scans from top to bottom like a CRT, illuminating a narrow moving band of lines as it does so [88]. A 1 ms capture of this is shown in Figure 5.3. The DFE system should have a visible latency equivalent to the maximum width (in time) of the rolling band, while that for the GPU system will be the time it takes the band to traverse the screen. Since the age of the oldest visible pixel is limited, the skewing features are not so pronounced in the 1 ms capture, although they can be seen with the help of grid-lines.

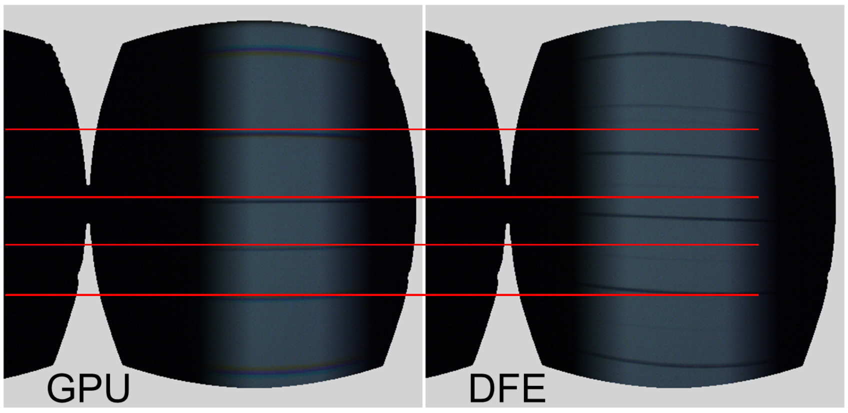


Figure 5.3: Captures of the DK2 screen with a 1 ms exposure, with grid lines to help delineate the skew in the DFE render due to motion

5.3.3 Image Fidelity Analysis

We use IQMs to measure the abilities of our rendering systems, comparing their true output with what they would ideally display, if we could build a system with zero latency. The expectation is that the measures will differ between the systems, showing that the rendering technique has a significant effect on what is perceived by the user.

5.3.3.1 Procedure

While previous studies have incorporated animation ([260]), none compared renders with moving viewpoints. With our renderers however, differences will only be apparent under motion. To create a suitable set of renders, we tracked the head motion of a human participant. We then selected a segment of the capture which had a range of angular velocities consistent with previously observed maximums under voluntary head motions [68]. With this tracking data we were able to produce a set of reference images, and drive the rendering systems in real-time with the same motion.

To measure the quality of the rendering systems' output, we required a set of ground truth reference images. These images are what would be displayed by an 'ideal' VE with zero latency. We produced a set of 18,000 images spaced 1 ms apart in time. Since an ideal system would update the entire display instantly, there was no point in producing images with a temporal resolution beyond that of the captured tracker data. For each image, its timestamp was determined and used to sample the tracker data by retrieving the nearest sample. This sample was used to configure the pose of the camera, then the frame was rendered on the GPU as if it were driving the HMD. Instead of being drawn to the display however the frame was read back from the GPU and written to disk. The result is a sequence of images showing what the HMD should display, if the system had an end-to-end latency of zero and was run at 1 kHz.

The tracking data was then used to drive both rendering systems in real-time, drawing to the screen of the DK2 while it was captured with a camera. The synchronisation LED was used to indicate when in the capture the first tracking data was read into memory. The frame at which the LED transition occurs is considered the epoch, and the timestamp of future frames relative to the first tracking sample are calculated based on the framerate of the capture. The synchronisation LED is cycled by the CPU at 1 Hz to ensure the clocks of the camera and the CPU were matched. Each rendering system was captured twice, once with a 1 ms exposure at a framerate of 257.7 fps, and once with a 13.2 ms exposure at a rate of 75 fps. This is equivalent to the framerate of the DK2 display and approximates a high persistence equivalent of the display. When in focus the DK2 screen and Arduino consumed an area of the frame 1280x600 and the display area of the DK2 screen was 874x491. Examples of the real-time captures are in Figure 5.4 and an example of a ground truth frame is in Figure 5.5.

Once the captures had been taken and synchronised, the area outside of the eye view-ports (which is invisible to the wearer of the HMD) was masked and set to black to avoid any luminance changes within it skewing the IQMs. For each tracking sample, the closest (in time) images were selected from the ground truth sequence and real-time capture and these images were compared with a number of IQMs. This was done for all 12,100 tracker samples, and repeated for all four captures (the DFE & GPU with 1 ms exposures (low-persistence) & 13 ms exposures (high-persistence).

The IQMs always operate on the entire image. This means the masked area outside of the view-ports will influence the measures but this influence will be constant across all the captures. For the pixels outside of the rolling-band, what is compared with the reference depends on the exposure time. For the 13 ms captures it will be the older pixels which were driven by the band previously, for the 1 ms captures it will be colour of the pixels when they are not being driven.

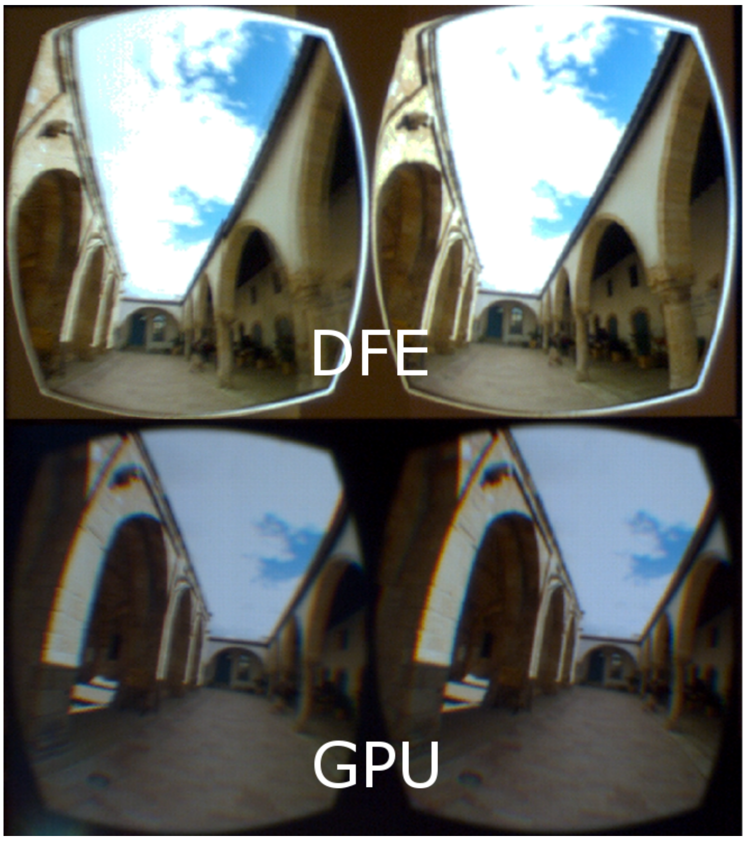


Figure 5.4: Example frames from two of the captures of the HMD screen (13 ms exposure)

5.3.3.2 Metrics

We cannot compare our stimuli with a real-world ground truth, because we do not have the ability to recreate the full dynamic range that the eye is sensitive to. However, we cannot assume either that the Image Quality Measures (IQMs) designed to mimic human IQAs can identify the optimal stimuli either. We therefore select a range of IQMs, of varying levels of complexity.

- **RMSE** is a measure of the absolute pixel-by-pixel difference of two frames.

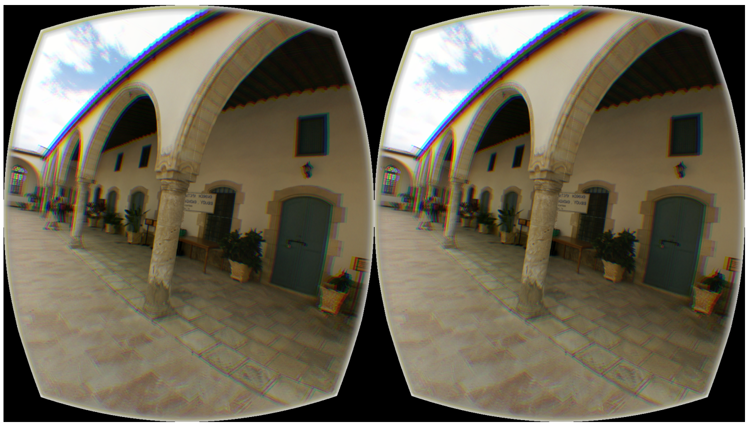


Figure 5.5: Example frame from the ground truth renders

It is a noisy measure, but simple and fast, and used in a number of previous works.

- **sCorrel (SCOR)** ([31]) performs a Spearmans Rank-Order Correlation on 8x8 pixel blocks. It is a more complex measure than RMSE, but is still not perceptually based. It has the advantage of being less sensitive to brightness changes and low frequency noise, both of which we can expect when comparing images captured with a camera to an offline render.
- **VIF** ([196]) compares the information available in an image to that in its reference. The information is extracted by passing the images through a 'distortion channel' approximating the Human Visual System (HVS).

We pick the above structural metrics because our investigation is geared towards spatial differences caused by scan-out and latency, whereas we can expect large colour discrepencies simply due to differences in the response of the rendering systems, the DK2 screen and the camera. Other frameless renders may pick different metrics that best reveal their differences. Another metric we would have preferred is sCIELAB [261], based on the CIELAB standard (Euclidean distance in L*a*b space). This metric is relatively simple and effective at attenuating low spatial frequency variations [31]. However it requires a mapping of RGB data to real-world wavelengths and dimensions of each pixel. We did not have this calibration for the DK2, nor the inverse through the camera, and so were unable to use it in this study.

5.3.3.3 Results

From the ground truth to the capture the renders were distorted considerably: by spatial distortions of the lens of the camera, differences between the FOV of the two rendering systems, luminance responses between the two systems and the colour responses of both the display & the camera. How significantly these differences affected the fidelity metrics was dependent on the content of the part of the scene that was visible, and therefore the orientation of the viewport.

This means that the absolute measures are not comparable between two systems, and the effects due to latency will be masked by the effects due to simply pointing the virtual camera in another direction. To ameliorate this, we normalised the error metrics and performed a multiple linear regression on the roll, pitch, yaw and average velocity values of the orientation (predictors) for the measures (responses). The intent being that the velocity coefficient would be mostly free of the influence of the changes due to orientation alone. This is as the velocity correlation will be performed on the residuals after these effects have already been accounted for by the model. The velocity predictor was derived by computing the average of the orientation component angles for each sample, taking the derivative, and then passing through a 10 sample wide smoothing filter.

We performed the linear regression for all captures at both exposures, and the results are shown in Table 5.1. Only significant predictors ($p < 0.05$) are shown. Each measure was computed for all tracker samples and so each model has 12100 observations and 12095 degrees of freedom.

An implication of considering each angle element individually for the absolute value, but averaged for velocity, is that the magnitude of the velocity coefficient cannot be compared with that of the angles, as the influence due to orientation will be distributed between these. The purpose of the models are to remove the influence of orientation however, not to examine it. More revealing are the changes in the velocity estimate between the conditions. Considering what we know about how the systems render, we can make four observations that are supported by Table 5.1.

1. The system with higher latency (GPU), should be more sensitive to velocity,

Coefficient Estimates and R^2 values for three IMQ multilple linear regression models ($p < 0.05$)				
Exposure Time	**1 ms**		**13 ms**	
Predictor	**DFE**	**GPU**	**DFE**	**GPU**
		Normalised Root Mean Square Error		
Roll	0.060	0.063		0.045
Pitch	0.404	0.430	0.210	0.295
Yaw	0.055	0.058	0.031	0.055
Velocity	1.010	1.089	1.008	1.531
R^2	**0.434**	**0.449**	**0.296**	**0.383**
		Spearman's Rank Correlation Coefficient		
Roll	-0.015	0.009	0.084	0.134
Pitch	0.034	0.044	-0.170	-0.201
Yaw	0.010	0.010	0.010	0.021
Velocity	0.039	-0.144	-0.490	-0.802
R^2	**0.192**	**0.215**	**0.436**	**0.481**
		Visual Information Fidelity		
Roll	-0.006		0.058	0.037
Pitch	0.012	0.004	0.031	0.012
Yaw	0.003	0.001	0.022	0.010
Velocity		-0.033	-0.715	-0.410
R^2	**0.155**	**0.142**	**0.236**	**0.108**

Table 5.1: Parameters for three multiple linear regression models for the RMSE, SCOR and VIF IQMs, showing only coefficient estimates with $p < 0.05$.

and we see this for almost all cases. The exception is the 1 ms RMSE model, in which we expect the difference in effect is hidden by the noise in the metric.

2. The more complex and sensitive to structure the IQM, the more it should reflect the differences in rendering approach. This is because they should be less sensitive to colour and luminance responses of the screen and camera, which we do not account for. We see this as SCOR and VIF have better fits than RMSE, and larger differences in the coefficients between the GPU and DFE, and 13 ms and 1 ms captures.

3. As the exposure time increases, there will be a higher number of 'older' pixels visible, which will result in a higher average error across the whole screen. This will be exacerbated by high velocities, where the discrepancy of these

older pixels will become more egregious. This is the case regardless of the underlying ‘tracker to beam’ latency, and is reflected in the 13 ms captures having larger coefficients than the 1 ms captures, for both the DFE and GPU.

The DFE should have an advantage in the 13 ms captures however. This is part due to the lower system latency, limiting the age of the oldest pixel, but also because the frameless nature of the display means the age of the oldest visible pixel increases at a constant rate equal to the scan-out time, whereas for the frame-based GPU it increases faster (scan-out time + time since the frame was rendered). The DFE typically has a smaller coefficient for the 13 ms captures, although to what degree this is due to the frameless nature, and what due to the lower average latency, we cannot say.

4. With a 1 ms exposure time, there will be fewer older pixels visible. As a result the error should be less dependent on the average latency across the entire frame, and more dependent on the ‘tracker to beam’ latency at any given time. This is what we see, with the SCOR and VIF coefficients being smaller for the DFE than the GPU - so far as to be statistically insignificant for the VIF measure.

The results show then that the stimuli produced by systems rendering the same VE, can vary significantly depending on which rendering approach was used under certain conditions (in this case high viewport velocity). IQMs attempt to quantify the perceived difference between two images. The coefficients of velocity are typically smaller for the DFE and low-persistance captures. This implies lower latencies result in higher fidelity VEs under user motion, and that the highest fidelity is provided by the DFE. While the IQMs are perceptually based, it cannot be said that the DFE provides a better experience in absolute terms. To begin with, the comparisons above are done at single points in time, with no consideration of the stimuli before or after. There may be significant temporal interactions with the HVS, which our experiment will not detect. Since measures are not directly comparable between the systems, we also cannot say that one system has generally higher fidelity than the other, only that one is better at approximating the ideal under motion. Further some artefacts may be desirable, such as blurring to reduce the perception of jitter during object movement.

5.3.3.4 Outliers

There is one outlier, and that is the velocity predictor shows a positive correlation with image fidelity for the SCOR metric. Even in a 1 ms capture, where the age of the oldest pixel will be no more than a few milliseconds, the correlation should at best be insignificant. The effect is very small (0.039), but significant (p = 0.0057) and the R^2 is low (0.192). We have no explanation for this result, and can only theorize that it is due to covariance with the orientation. For completeness, the mean covariance for all conditions was $< 1e^{-4}$ for SCOR & VIF, and $< 1e^{-3}$ for RMS.

The DFE estimate for the VIF 13 ms capture is also smaller than that for the GPU, which may on first glance be surprising but should not be considered an outlier. The DFE system maintains its low latency at a cost of image distortion, as it is skewed during scan-out under motion. VIF is the most complex measure and may consider this distortion more egregious than the discrepancies in orientation due to time. When the number of visible (older and distorted) pixels is reduced (in the 1 ms capture) this effect disappears. Future studies may be improved by designing new metrics specifically for frameless renderers, or constraining existing ones only to operate on the visible regions of low persistence displays at a given point in time.

5.4 Discussion

Ray casting is a highly constrained subset of ray-tracing, making it amenable for hardware acceleration such as we have done. The configuration is similar to that of Regan & Pose's Address Recalculation Pipeline, where the light transport is simulated externally and the device computes novel views at a high rate, in an operation analogous to image warping. Unlike the Address Recalculation Pipeline however, we use ray-casting with simple geometric proxies. This allows our renderer to operate standalone while still permitting users to translate, as well as rotate, in the virtual world. A 2D image warper in this case would introduce spatial distortions. Users will still not see correct view-dependent visual effects such as specular highlights however - for this our system would need to be coupled to a renderer re-generating the maps in real-time. In this case there would be two latencies, for different visual effects, to consider. This is a similar situation to that encountered with dynamic objects, the behaviour of which may be computed by a loop with latency characteristics quite different to that of the rendering loop.

We proved the latency of our system is no more than ~1 ms, although the latency of the rendering stage itself is likely much lower. Typically, latency is considered as a discrete characterisation of a constant delay between user input and the response of the display - but this is not complete. Latency changes across the display during scan-out, and is a function of a number of things, including the time to render a single frame, the pipeline depth between the renderer and the display, and the scan-out period itself. Different renderers do not have equal latency responses, and changing the rendering approach can significantly alter this response. For example, we show how the profile of the frameless render is very different to that of a GPU. The latency of the frameless renderer being lowest at the location of the scan-beam, instead of increasing with it from the top of the screen, as with a frame-based system. This is because the frameless system has a ‘tracker to beam’ latency lower than the frame period, so the more time elapsed since a pixel was driven, the higher the latency of that pixel. Conversely frame-based systems scan out discrete frames, synchronised to the top of the display. As soon as a frame is finished, it is ageing even before scan-out begins, therefore the delay increases as the scan-out proceeds. This difference is manifested as a skew feature under motion on the frameless renderer, as a single scan-out is an amalgamation of multiple tracker samples. Using objective IQMs, we assessed abilities of each system to faithfully recreate a virtual world. Unsurprisingly we found the system with the lowest latency performs better than the high latency system. Further though, we show how the rolling scan approach to low persistence on the DK2’s OLED screen reduces the effect of velocity on fidelity. The effect is reduced for both the GPU and the DFE, but on the DFE it is reduced to practical insignificance. We have shown that the frameless renderer has a higher fidelity under motion than traditional approaches. However, how this affects participants in a real

VE system is not obvious. For example, one implication of the different rendering techniques is the relative latency of the eyes. In the DK2, the screen is mounted horizontally, so when driven by a frame-based system one eye will always have a higher latency than the other. On a frameless system the point of lowest latency is constantly changing, meaning that on average the eyes could have an equal latency.

In order to perform the evaluation we disabled the timewarp functionality of the GPU apparatus' CPU application. In other VEs cases though the Oculus SDK can combine two features to reduce apparent latency significantly. The first, View Bypass, compensates for the rendering latency. The GPU compensates for lens distortion by rendering the scene to a typical (i.e. planar) viewport, then texture mapping the render to a mesh which counteracts the distortion of the lenses. The GPU renders this mesh to the viewport displayed to the user and in doing so applies a post-rendering warp. View Bypass involves re-sampling the tracker right before performing this second render, identifying the change in tracker state since the original frame was rendered, and compensating for it by warping the 2D image and/or the distortion mesh itself. Regardless of the complexity of the original render, the distortion process remains the same for each frame, making the time to complete it highly predictable. This facilitates the second technique, Time Warping, in which the post-rendering warp is left as late as possible so it completes just in time for the next scan-out to begin. The intent is to reduce perceptible latency. It has the disadvantage though of introducing unpredictable spatial distortions. Further, while the perspective may change, dynamic scene content cannot be warped in such a manner. With View Bypass and Time Warping, it is in theory possible to warp on a line by line basis, essentially implementing a frameless renderer. A naive implementation on a GPU however would issue thousands of draw calls per frame (one per line) and the drop in frame rate on any typical PC due to CPU overhead would likely negate any gains. Such a solution is also restricted to per-line warping, whereas a frameless renderer such as ours can update on a per-pixel basis.

Our results were supportive of our observations of the behaviour of the systems, but our experiment had a number of limitations. First, we did not account for the colour responses of the screen or the camera. This lead to high noise floors for the simpler measures such as RMS. Perniciously though, it also meant that the magnitude of the reported errors were dependent on absolute orientation. We used multiple linear regression to minimise this influence, before examining how the render fidelity varied with velocity, although there is no escaping that velocity is in part a function of the orientation.

We did confirm that the covariance of the predictors was minimal for our models, but are still unable to explain the outlier described in Section 5.3.3.4. In expectation of such outliers, and in recognition that the response of participants to visual stimuli is not entirely understood, we chose to use multiple inherently different IQMs. The more sensitive to structure the IQM, the more sensitive to velocity it appeared to be. However this also revealed that in the case of high persistence displays, that for frameless renderers such as ours the skew feature may be more egregious to a participant than the discrepancy due to latency. In the future, a better way to perform such assessments may be to capture the ground truth from the camera and screen. To do this, the CPU application could be modified to draw a static image (pertaining to one tracker sample) to the screen. After enough time for all the pixels to transition has passed, that frame could be captured and considered as what an ideal, in terms of zero latency everywhere, display would show. Capturing the ground truth in such a way would reduce the noise floor to that inherent in the sensor and display itself, removing discrepancies due to colour and slight differences in the geometry transforms.

5.5 Summary

We evaluated the performance of our low latency real-time ray caster, while interoperating with an existing, readily available HMD. Using a series of high and low speed video captures, and objective IQMs, we investigated the implications of combining a frameless renderer with a sequential scan-out OLED display, and compare this with an equivalent system, but built with a GPU. The results were generally as expected, with the apparatus capable of perceptibly responding to user input more quickly showing higher fidelity. We chose a range of IQMs because there is still ambiguity in how to judge the perceived similarity of two images by a person. It may be that there is no correct model as the measure depends too much on the level of perception being examined and the context in which it is done so. IQMs however have proven to be versatile and practical measures for comparing two systems however. Further many more IQMs are available, as well as analogous techniques such as fidelity metrics based on user perception, that could be used in future studies.

The use of ray casting, rather than simpler affine transforms, allows our renderer to draw relatively complex virtual environments. The next step will be to put users in a system built with the frameless render to see what effects, if any, there are on presence or other performance measures of a VE.

This is as far as the rendering system was developed. For the experiment described in Chapter 7 the renderer was integrated with an advanced tracking system, and additional worlds were created for it to drive. The architecture and capabilities however remained unchanged. In Part II, user studies into the effects of latency at high and low levels using the novel renderer are reviewed.

Part II

The Effects of Latency

Chapter 6

The Effects of Latency on Physical Interaction

Advanced graphical interfaces are commonly used to facilitate intuitive visualisation and manipulation of data as efficiently as possible. Some do this with abstractions such as widgets or manipulators. Others, such as pseudo-physical or skeuomorphic interfaces, exploit knowledge about natural object behaviour to allow more intuitive interaction techniques. For example, by constraining the behaviour of virtual objects so they obey the laws of physics [89]. Such approaches are often used in synthetic environments, with the explicit goal of establishing the sensorimotor loop.

In Section 2.2.2, we reviewed previous studies of the effects of latency on a number of sensory modalities, from latency detection in immersive virtual environments, to its effects on indirect physical interaction. The latter received considerable attention due to its ubiquity and importance, with many previous studies using motion primitives such as pointing tasks to investigate the effects of latency [129, 240, 36, 221, 168]. Only recently though has it become practical to build apparatus with latencies low enough that the limits of its effects may be found [99].

In Part II, we present two studies of the effects of latency. The first is concerned with the effects on low latency physical interaction.

The human motor system has been modelled as a control loop, with inherent delays that place natural limitations on performance; movement cannot be coordinated on time-scales smaller than the inherent delay [15]. We therefore hypothesized that there may be a non-zero external latency which has no perceptible effect on the sensorimotor loop. Latency cannot be removed given current technology, but we can compensate for it. By understanding how latency affects the different modalities that create an effective user interface, we can distribute resources of computer systems to minimize negative effects and create a better user experience.

We investigated the modality of indirect physical interaction, using familiar desktop based pointing and steering tasks. We used our prototype 2D sprite renderer to create an apparatus similar to previous studies [129, 221, 168] but capable of much lower latencies. Our results were unexpected and have significant implications for future studies using physical tasks to investigate latency, as we show that considering only total movement time and not its constituent parts may result in inconclusive measurements which hide the effects of latency.

6.1 Survey of works on physical interaction

The chief concern in facilitating natural physical interaction in synthetic environments is enabling the formation of the sensorimotor loop. As we have seen, this underlies presence and the ability to induce experiences. As important as enabling the sensorimotor loop is, natural physical interaction has been of interest in many aspects of Human-Computer Interaction, both inside and outside of the synthetic environments. This interest is partly is due to the importance of improved performance to justify the interface. For example, Smith et al. [213] applied real world physical constraints to the objects in a 3D editor, decreasing the degrees of freedom of the objects in a familiar way. Users showed improved performance when interacting with the editor using 2D interaction techniques. The interest is also partly due to the ubiquity of such interfaces. Even when abstractions are present, actions are still predominantly basic motion primitives such as reaching and pointing [89]. Accordingly, there are many works in this area which can be applied to synthetic environments.

6.1.1 Models of the Motor System

A number of authors have constructed theoretical models to explain the operation of the visuomotor system. One such model is that of Botzer & Karniel [23]. The authors derived their model from observations of delay compensation behaviours. Participants performed Fitts's law style tests [56]. They were allowed to adapt to different latency conditions, and then the visual feedback was removed and at the same time the latency changed. By observing how user motion changed under this new condition, the authors tested where in the hypothesized control loop delay compensation was performed. Whether in the feedforward model, which plans the trajectory, or the feedback loop, where correction commands are issued based on visual feedback. Overshoot and undershoot were present in reaching tasks in unexpected delay conditions. This demonstrated dominance of an adapted visual feedback stage over the feedforward planning stage. They also found that while discrete reaching movements returned to baseline conditions (that is, the users no longer overshoot or undershoot), rhythmic ones do not. This suggests there is adaption in the forward model, but it is dependent on movement type, leading to their model incorporating multiple pathways.

Beamish et al. [15] considered the motor system as a Vector Integration to Endpoint (VITE) circuit. In the VITE circuit a continuous outflow of commands to the muscles are a result of the motor system attempting to reduce the difference vector between the intended target position and the present position. The commands are generated by the neuron population calculating the difference vector, based on the present position estimation from a population which integrates all previous movement commands. They note the VITE circuit as one of the earliest models to suggest how the movement characteristics described by Fitts's law are a result of underlying neurobiological mechanisms. The authors introduce time delay between the two populations into this model. By drawing comparisons with a servomechanism model, they show that for a system to be stable, the gain (magnitude) of the movement commands must be below a value which is a function of delay. It should be noted that the model described above does not take into account visual feedback - or indeed any external delays. That is, even considering a system based only on proprioceptive cues the authors demonstrate a hard upper limit on performance.

Beamish et al. [14] pursued this model, using it to estimate the inherent effective feedback delays in the motor system based on the results of previous Fitts's law style experiments. They expressed the performance of the VITE circuit (movement time) in terms of difference vector neuron population time constant, and feedback delay. They could then relate these parameters to the observed Fitts's law constants *a* & *b*. Using the measurements available from over 25 previous Fitts's law style studies, they found feedback delays between 0-112 ms, generally below 60 ms. They also found that the nature of the VITE circuit imposes a limit on the performance of unidirectional movement. When this limit is expressed as a Fitts's law Index of Difficulty (ID), it happens to be the typical range employed by previous experimenters.

6.1.2 Fitts's law

Most studies on physical interaction, such as that of Jay et al. [91], use Fitts's law style tests. A good review of Fitts's law is by Seow [194]. Fitts's law is an emergent property rather than a description of the motor system operation. This is discussed by Bootsma et al. [22] and Huys et al [81]. Both sets of authors demonstrate that by observing the patterns of motion directly under different conditions, Fitts's law is a good summary of complex motor processes. However there is increased asymmetry in the amount of time spent in the acceleration stage compared to the deceleration stage as latency increases. The pattern of movement is significantly different between rhythmic and non-rhythmic movement, and as ID increases rhythmic pattern becomes more like the discrete pattern. This suggests multiple functionalities acting in parallel, such as in the model proposed by Botzer & Karniel [23]. Botzer & Karniel referred to Rythmic/Non-Rythmic as Slicing and Reaching respectively.

As a characterisation of the motor performance, Fitts's law has been observed a number of times under a range of conditions. Its repeatability and invariance make it valuable for testing the effects of various factors on user interaction. For example, Adam et al. measured the difference between egocentric guided movement and allocentric guided movement [2]. This was expanded on by Blinch et al., who found that the most significant effects occurred between the presence of allocentric markers and the preparation stage of movement [21]. Perrault et al. tested the scale effect using Fitts's law [161]. Jax et al. tested the effects of obstacles in the movement path [90].

6.1.2.1 Fitts's law and Latency

For the same reasons described above, Fitts's law has been used extensively to investigate the effects of latency. MacKenzie & Ware did one of the first studies in this area, reformulating Fitts's law to account for additional movement time delay [129]. They estimated the base latency at 8.3 ms, and between 16 and 225 ms of latency was added. Pavlovych & Stuerzlinger suggest that the base latency could actually have been ~60 ms though [168]. Performance began to decrease significantly at the 75 ms condition. Ware & Balakrishnan used 3D reaching tasks in order to compare the effects of hand tracking delay with head tracking delay in an immersive Virtual Environment. They tested latencies between 87 and 337 ms. Teather et al. measured the effect of latency and jitter on performance in Fitts's style 2D tasks, and 3D object movement tasks, while looking for an effect of the type of tracker used. They measured the latency of their system at 73 ms and found that the performance degradation was equal for the tracker devices [221]. Pavlovych & Stuerzlinger performed a Fitts's law style test to determine the effects of jitter and latency. They found a strong interaction with latency and jitter. Further, with low jitter the effects of latency were dominant, but the jitter degraded performance at a higher rate than latency. The authors measured the base latency of their system at 33 ms, and added up to 100 ms [168]. Chung & So considered that latency may affect the stages of movement differently. They studied the effects of latency in Fitts's law style tests but on target width and distance separately. There was strong interaction between latency and target width, but not target distance [36].

6.1.3 The Steering law

There is evidence ([23, 15]) that the motor control system consists of multiple complex elements, some acting in parallel, and that the effects of latency on these is not equivalent. Thus in our experiment, aside from a Fitts's law-style task, we introduce a second task based on the Steering law. It is designed to exercise the real-time correction functionalities predominantly and force the user to continually change goals as they move.

The Steering law was introduced by Accot & Zhai. It was originally derived from Fitts's law, considering a path as a sequence of goal crossing tasks. The completion time was the measure of performance, and was estimated to be the sum of the time to complete the individual goal crossing tasks, that make up a path [1]. It was extended by Kulikov et al., who used the concept of effective width to demonstrate that the Steering law was even more accurate than originally shown [109].

Like Fitts's law the Steering law has been used to investigate the effect of specific factors on user performance. Liu et al. investigated which path properties affected user performance. The path properties considered were curvature and width [123]. Liu & Liere continued to investigate the effect of these properties changing within a path. In their test the path was presented as a tube. Participants were encouraged to remain within it by pushing a ball through it with the cursor. We model our implementation of the Steering law task on theirs. On examining the user movements, they assert that the behaviour does not resemble a goal crossing task, as much as a set of small ballistic movements [121].

Pavlovych & Stuerzlinger investigated the impact of latency on tracking tasks. This task is analogous to the Steering law task. The authors point out however that the Steering law itself does not apply. This is because there are no boundaries to movement outside of the target area and the user is required to correct velocity as well as direction. The experimental setup had a base latency of 20 ms, and an additional latency of 30-150 ms. The authors observed a significant effect of latency on tracking accuracy, and that it was not symmetric: users had a smaller error perpendicular to the target, than tangential. The latencies tolerated before a significant interaction became visible were higher than in previous studies (50 ms for latency and 40 ms for jitter). Another interesting observation was that performance decreased for the condition with the lowest additional latency (20 ms), improved between 20-50 ms, then for latencies above 50 ms degraded again but at a slow rate [169].

6.1.4 Investigation of very low latencies

The closest study to that described here is that of Jota et al [99]. They studied the effects of latency on direct interaction surfaces, with their High Performance Touch prototype - a touchscreen with a latency of less than 1 ms. A number of previous studies have investigated the effects of latency on direct touch interaction, but none at such low levels. Participants performed Fitts's law style tests. Of particular interest in this study, is that the user received visual feedback from both their non-latent hand and the latent cursor simultaneously. How the potentially conflicting stimuli affect performance is not clear. Participants showed a range of behaviours in response to the latent cursor, from ignoring it completely, to leading it, to slowing their movement so that it remained under their finger at all times. The additional latencies were between 1-50 ms. The authors reported no observable difference in performance between latencies of 1 ms and 10 ms. A linear regression fit suggested the performance floor may not exist. By segmenting the movement into stages, the authors demonstrated that the effects of increasing latency on these are not symmetric, as Chung & So and Bootsma et al. showed for increasing ID [36, 22].

6.2 Experiment

A number of studies have used performance in motor tasks to detect the effects of latency. Few though have investigated latencies at very low levels. Jota et al. found a potential floor for direct interaction tasks [99]. Indirect interaction techniques however remain important for both 2D and 3D interfaces. They can exceed direct interaction in both efficiency and precision [89]. We therefore continued the investigation into indirect interaction.

6.2.1 Apparatus

To conduct the investigation an interface with very low controllable latency was required. The indirect input Fitts's law and Steering law tests require a 2D interface. The participants interacted through a cursor, which had to respond to the user within the shortest amount of time possible. We began with our prototype sprite renderer from Section 3.6. We combined this with a high-speed (120 Hz) LCD monitor. At

~1 ms, the latency of our system from input to video signal output is much lower than previous apparatus. We are limited by display technology however. The display scan-out time increases our end-to-end latency to 6 ms. Beyond this, persistence of the image on the monitor can cause the perception that latency is greater than the average frame period. This is because the stimuli at any time is a blur between the current stimuli and the previous one. We selected a highly responsive, high frame rate monitor (an ASUS VG248QE), minimizing perceived latency due to both scan-out delay and persistence. The limitations in available display technology are shared by previous authors. Out of the aforementioned studies only Jota et al. secured a better performing display than the VG248QE. They did this by building a custom display based on a Digital Micro-mirror Device driven in a very low chromatic range [99].

6.2.1.1 Tracker

We used a number of input devices in pre-trials, including a mouse, Phantom Geomagic Touch (formerly Sensible Omni)[1], and Wacom drawing tablet. The performance profiles of the pre-trial participants were ostensibly consistent so we ran the experiment with the mouse as this was the most comfortable device. The mouse was a Kingston Mouse-in-a-Box optical mouse, with the Control-Display gain set to 1. It was sampled at 1 KHz. To prevent conflicting visual cues, the users hand was obscured from view using a black cloth. The complete apparatus is shown in Figure 6.1.

6.2.1.2 Latency

Once the apparatus was integrated, we used the parallel port of the host computer and an output from the DFE to probe the latency of the rendering stage of our system. The DFE illuminated an LED on receipt of a specific input. High speed video monitored the input device, and the LED. This arrangement was chosen as it allowed us to monitor both the input device and the scan-out of the display, with no further instrumentation. The latency between the input and the LED was below the temporal resolution of the video (1 ms). In the best case scenario the user begins just prior to the cursor is drawn. In this case the latency is between 1-2 ms - predominantly the

[1]http://www.geomagic.com/en/products/phantom-omni/overview

mouse sampling time. In the worst case the user moves immediately after. In this case the latency is 8-9 ms. This is the mouse sampling time (1-2 ms), the rendering time (<1 ms) and the period of one frame on our display (6.9 ms). We expect the latency to be ~5 ms on average. We measured the total end-to-end latency of our apparatus using the cross-correlation variant of Steed's Method [62]. Correct operation of the apparatus was confirmed by measuring the latency throughout the investigation, between each participant. The baseline latency was measured at 6 ms, with the tolerances described for the measurement method [62].

6.2.1.3 Processing and Transport

With the exception of the novel renderer, our system was a typical desktop, based around an Intel Core i7 PC and running CentOS 6. The tests were implemented in a thread running with real-time priority, controlled by a non-realtime manager application. The renderer was accessed using Maxeler's low-latency API for communicating with the DFE via the PCIe bus. Like the mouse access, this makes use of polling, rather than events. The real-time thread communicated with the managing application via flags in memory. We profiled the thread to ensure that we only used calls which would not cause it to yield unintentionally. The thread was given the highest priority. The result was that the thread was never pre-empted, and latency due to time-slicing of the CPU was not introduced.

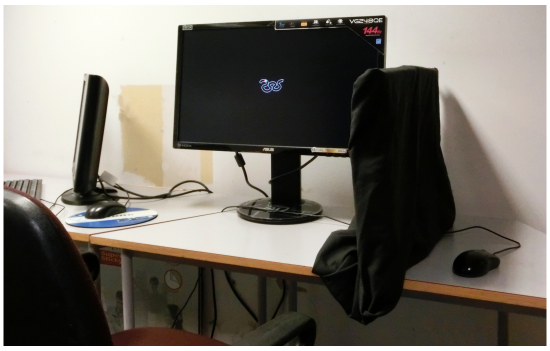


Figure 6.1: Experimental apparatus that the participants interacted with

6.2.2 Participants

30 participants (19 M/11 F) with an average age of 27 (Standard Deviation: 4 years) from within University College London were recruited for the study. Participants

were paid £5 for taking part.

6.2.3 Procedure

Participants typically sat ~0.6m in front of the display. They were invited to move the chair, display and mouse to make themeslves comfortable. The mouse and screen were moved between hands if requested. Once comfortable, they were shown the two tasks and allowed to practice each for as long as they wished. All participants were instructed to move as fast as possible. The participants spent 20-30 minutes completing the actual tests. The time to complete the whole experiment was 30-50 minutes. Our experimental design is very similar to the one-directional tapping task described in ISO9241-9 [84]. We deviated by having users make discrete movements, rather than repeated rhythmic movements. This is because the motor system behaves differently during these two types of motion [81, 23]. Further, the seminal works using Fitts's law to investigate latency, such as that of MacKenzie & Ware [129], use discrete tasks.

6.2.4 Tasks

6.2.4.1 Fitts's law

For the Fitts's law style tests, participants saw a box on the screen ~2cm x 2cm, which remained throughout all the tests (the staging area). Clicking on this box would start the test, and a target would become visible to the right. Participants were instructed to click on the target as fast as possible, then in their own time move back to the staging area. Clicking the staging area a second time would begin the second test, and they were to repeat this until all tests were complete.

6.2.4.2 Steering law

For the Steering law tests, users were presented immediately with a 2D path, and at the start of the path, a green ball. They were instructed to push the ball through the path, by placing the cursor behind the ball and moving it forward through the path. Users were again told to maximise speed, and were told that keeping the cursor within the path would be the fastest way to complete the tests.

Examples of the stimuli seen by the users are in Figure 6.2.

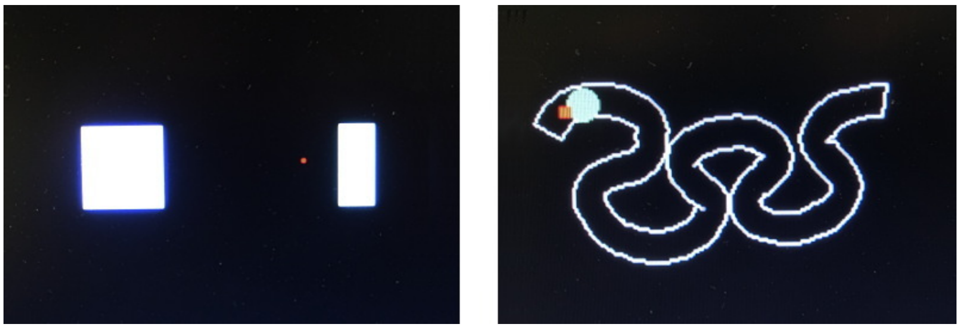


Figure 6.2: Images of the stimuli the participants were exposed to

6.2.5 Design

The experiments had three independent variables: Latency, Width and Distance (Fitts's)/Curvature (Steering). For both Fitts's law and the Steering law there were four conditions of spatial difficulty, summarised in Table 6.1. For each condition there were six additional latencies (0, 10, 20, 30, 50, 80) for a total of 2x2x6 (24) unique conditions. Unique Fitts's law conditions were repeated 8 times, and Steering law conditions 5 times. The repetitions were averaged for each participant, resulting in 720 data points for the Fitts's law tests and 720 for the Steering law tests. The tasks had low entertainment value, and fatigue was a concern. Since we expected the effect to be small, we optimised for a high number of latencies and repeats at the expense of spatial difficulty range. The widths and distances were informed by pre-trials. The range of IDs found by these matched those of MacKenzie & Ware, and those estimated by Beamish et al. [129, 15]. The IDs were calculated using MacKenzie's method [127].

The Steering law paths were manually created, with one designed to emphasize sharper higher rate turns (predominantly exercising the wrist) and the other sweeping turns to exercise the upper arm (classed as curvatures 2 & 1 respectively). The IDs were calculated with Accot & Zhai's method [1]. The curves are not produced from any predicable function. This was deliberate, to prevent any unanticipated motor process (such as that used for reciprocal movement) from hiding the effect of latency on the on-line correction processes. Conditions were distributed to maximize the difference between sequential latencies. Within this constraint the widths and distances/curvatures were distributed randomly. All participants received the same conditions in the same order.

Condition			Index of Difficulty
		Fitts's law	
	Width (cm)	**Distance (cm)**	
1	0.25	4	4.09
2	0.25	11	5.49
3	0.9	4	2.44
4	0.9	11	3.72
		Steering law	
	Width (cm)	**Curvature**	
1	0.4	1	45.37
2	0.4	2	50.63
3	0.7	1	25.92
4	0.7	2	28.92

Table 6.1: Spatial Difficulty Conditions for both types of task

6.3 Results

6.3.1 Pointing Tasks

We measure Movement Time (MT) to be from the time the user clicks the staging area, to the time they click the target. Figure 6.3 shows MT for each of the latencies. As expected MT increases with ID, with a jagged appearance due to the small number of spatial (Width & Distance) conditions that do not have overlapping IDs [168]. We clearly see an increase in MT with high latencies, but not for low latencies. This is better illustrated in Figure 6.4 which shows how MT changes with latency for each condition.

6.3.1.1 Comparison with Previous Works

Studies conducting experiments most comparable with ours include [129, 221, 168]. All studies included Fitts's law style tests using mice, with latency as the independent variable.

- MacKenzie & Ware [129] investigated latencies estimated to be between 68 - 315 ms [168].
- Teather, et al. [221] investigated latencies measured at 35 - 255 ms.
- Pavlovych & Stuerzlinger [168] investigated latencies measured between 33 - 133 ms.

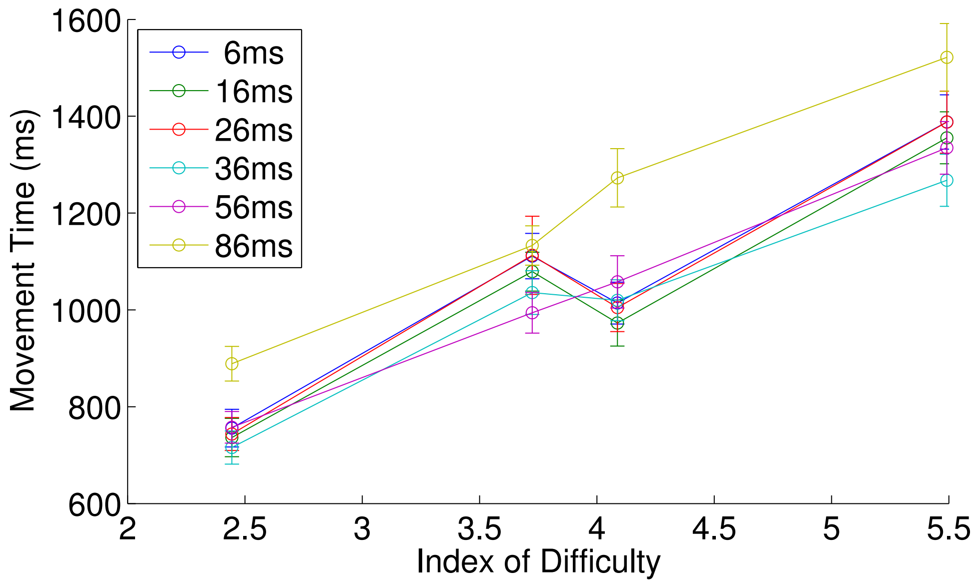


Figure 6.3: Movement Time against Index of Difficulty, for all latency conditions. Error bars indicate confidence intervals at 95%.

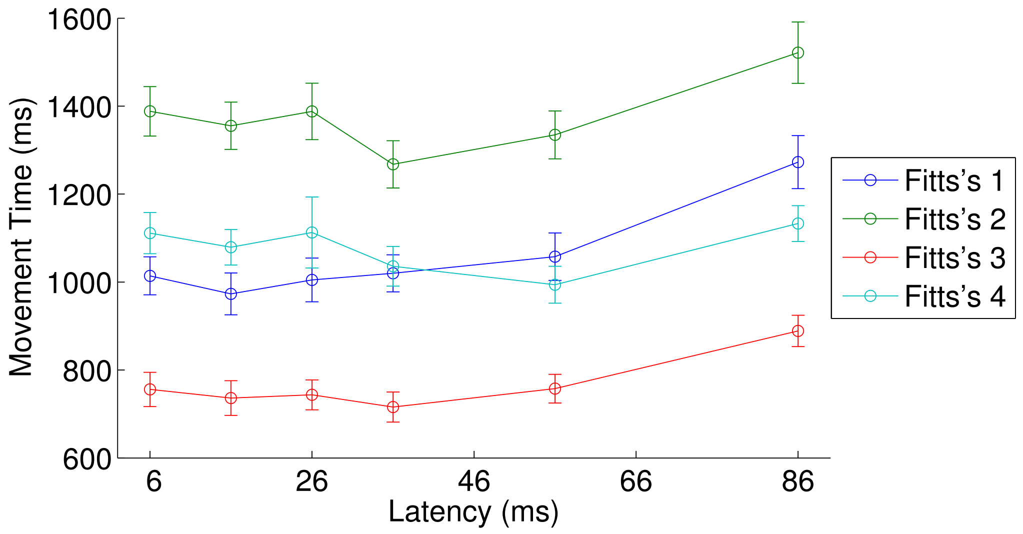


Figure 6.4: Movement times for each condition against latency. Error bars indicate confidence intervals at 95%.

For [221] and [168] the latencies measured are the total end-to-end system delay, the same as measured by us. We first consider only the higher latency conditions (36, 56, 86 ms) which are directly comparable with the previous studies above.

We fit a model using multiple linear regression and show a significant interaction with width ($\beta = -499.81, t(356) = -22.22$, $p < 0.001$), distance ($\beta = 37.48, t(356) = 17.95$, $p < 0.001$) and latency ($\beta = 4.02, t(356) = 11.30$, $p <$

0.001). We then fit MaxKenzie & Ware's model to our data and show an almost identical R^2 value (0.995 (ours) vs. 0.967 (theirs)). Finally we perform a one-way ANOVA as done by Teather, et al. and Pavlovych & Sturzlinger showing a similarly significant interaction $F_{2,357} = 21.32$, $p < 0.001$. All studies showed a significant almost identical multiplicative effect of latency with ID. We show the same effect for the overlapping latency conditions in our study. This is illustrated in Figure 6.5, which compares our results with those of previous studies. At lower difficulties our results appear slightly higher than previous works. The relationship though is identical, and our results are well within the inter-study variance. Thus for the higher latency conditions our experiments reproduce previous results.

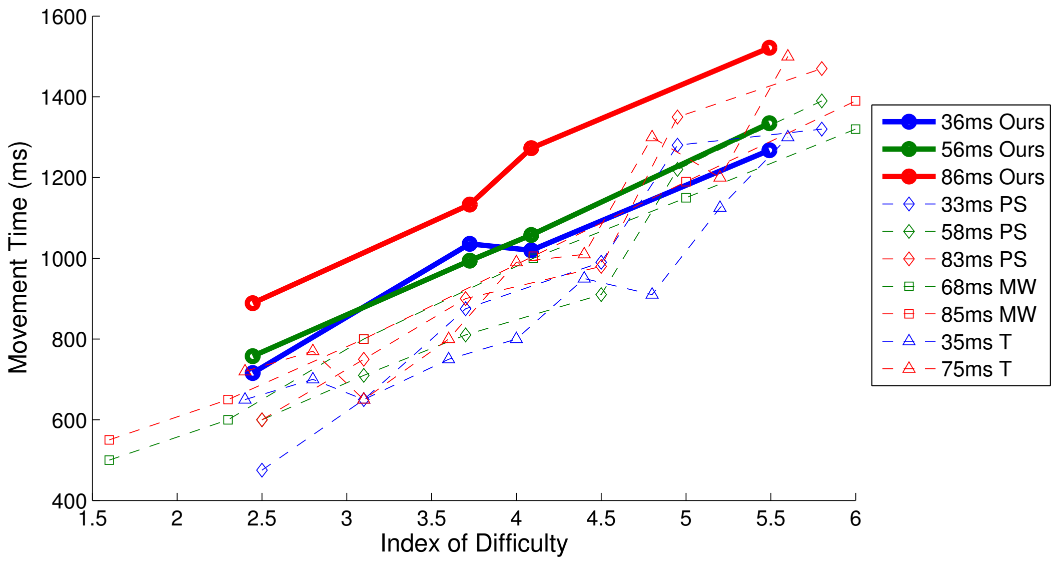


Figure 6.5: Movement time for the 36, 56 & 86 ms conditions of the current study, compared with the overlapping conditions from Pavlovych & Stuerzlinger (PS) [168], MacKenzie & Ware (MW) [129] and Teather, et al. (T) [221]

6.3.1.2 Deviation at Low Latency Conditions

When we consider the low latency conditions, our results begin to diverge from the expectations of ourselves and other authors. Considering only the lower levels of latency (6, 16, 26 ms), multiple linear regression demonstrates no significant interaction between movement time and latency ($t(356) = -0.27, P = 0.78$) and neither does ANOVA ($F_{2,357} = 0.48, P = 0.62$). We hypothesised a non-linear response as latency decreases. As shown in Figure 6.4 though, the linear relationship dissolves at higher latencies than we would expect. A clear correlation between MT and latency does not form until the latencies reach 50-80 ms, while studies have found an effect at far lower levels [99]. Further, in some cases user performance appears better in high latency conditions than in low latency conditions.

There have been hints of this effect in previous studies. In Fitts's law tests by MacKenzie & Ware [129] and Teather et al. [221] there were IDs at which users had near identical performance at two different values of latency. The differences were slight though and the equivalence could be argued to be measurement error. In pointing tests by Pavlovych & Stuerzlinger [168] the effect is more pronounced, with the 83 & 33 ms conditions appearing to alternately outperform each other depending on the ID. In a target tracking test [169] users had a reduced tracking error at 50 ms compared to 25 ms. Although this interaction was proved not statistically significant, the authors suggested that it could be caused by the users overcompensating and moving in front of the target at lower latencies. Another explanation is that users are more familiar with computer mice having latencies around 50 ms.

Until now these anomalous results have not warranted further investigation. Our tests however show a pronounced and repeatable effect. Observing the behaviour of the user during the task more closely reveals a possible cause. We suggest it is a result of the independent affects of latency on different stages of movement, happening at levels well below those at which performance supposedly improves.

While movement time is a useful metric, it does not allow for appreciation of the underlying processes. A number of works have hypothesised the motor system as a feedback loop, with an initial impulse followed by some form of continuous control. One way to characterise this has been to examine the symmetry of movement velocity profiles around the point of peak-velocity. As task difficulty increases so does the proportion of time spent correcting movement in the second - deceleration - stage [51]. An example is given by Bootsma et al [22]. We show that this deceleration stage may be further subdivided, into what we term the acquisition, and correction stages (defined below). Further, the impact of latency on these is not symmetrical.

Various parameters of kinematic profiles have been examined to gain insight into motor system functionality. There are different schemes to partition kinematic profiles. Partitioning based on peak-velocity is one example [51]. Another is that used by Meyer et al to partition motion into a primary movement and optional correction sub-movements for the two-component motor system model [145].

Bootsma et al previously used the peak-velocity scheme to quantify the effects of Fitts's law test parameters on the kinematic profile [22], and it has been used to investigate multiple theories of motor system operation [51]. Examining the effects of latency on the kinematic profile with respect to specific motor system models such as Meyer et al's however may provide new insights and is a subject for future work. For this investigation we partition the aiming motion into three stages:

Acceleration The time between the user beginning to move, and reaching their peak velocity.

Acquisition The time between the peak velocity and the user first reaching the target.

Correction The time it takes the user to settle and complete the task once the target has been reached.

Under very low latency conditions the majority of the time is spent in the acceleration and acquisition stages, so the correction stage is typically the time it takes the user to click the mouse button. Under high latency conditions the user overshoots and so the time in this stage is extended. The breakdown of the total MT into stages can be done by defining kinematic markers (e.g. the sample with peak-velocity) and using the position and timing data in the log files. The breakdown is shown in Figure 6.6.

As would be expected of a pre-planned impulse, multiple linear regression shows a strong interaction between the acceleration stage and distance ($\beta = 8.5, t(687) = 30.02, P < 0.001$), but not width ($t(687) = 0.97, P = 0.33$) or latency ($t(687) = 1.27, P = 0.2$). Performing multiple regression on the acquisition and correction periods independently, show the effects of latency are strong but

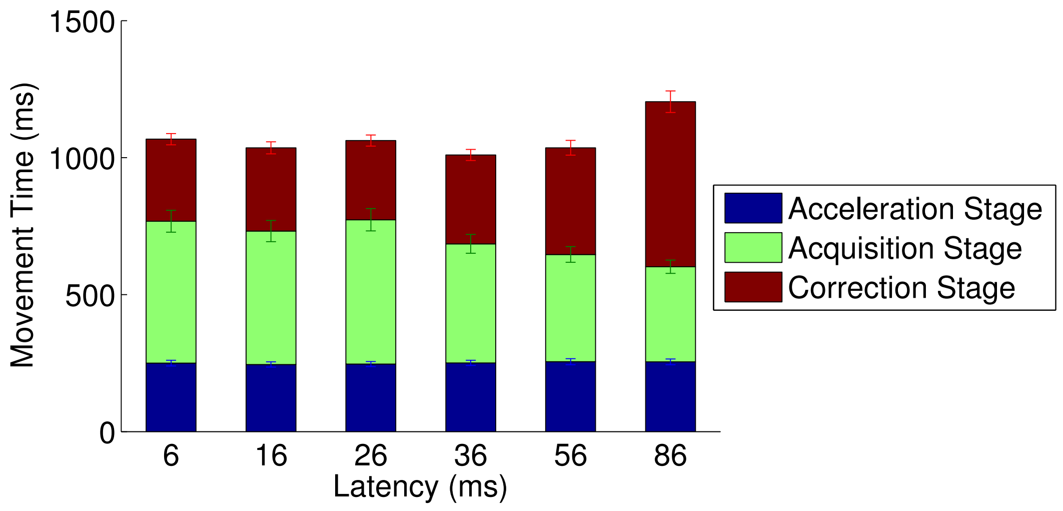


Figure 6.6: Mean time in each task stage, for each latency. Error bars indicate confidence intervals at 95%.

asymmetric. The results are shown in Table 6.2. The R^2 values for the acquisition stage and correction stage are 0.832 and 0.682, respectively. The error degrees of freedom for both is 687.

Variable	**Stage**			
	Acquisition		**Correction**	
	Coefficient	**P-Value**	**Coefficient**	**P-Value**
Width	-181.82	<0.001	-243.09	<0.001
Distance	41.28	<0.001	-6.06	<0.001
Latency	-2.25	<0.001	3.19	<0.001

Table 6.2: Multiple linear regression results for separate stage movement times.

As latency increases, the time in the acquisition stage decreases. This is accompanied by an increase in average velocity for the stage. That is, the user covers the same distance during this stage as before, but makes the motion in a shorter amount of time. Conversely, time in the correction stage increases. Total movement time is the sum of all three stages. Since the correction time typically increases faster than acquisition time decreases, higher latencies generally result in higher movement times. The effect on correction time is non-linear however, with large increases occurring only at high latencies. Therefore there is a subset of latencies, within which acquisition time decreases faster than correction time increases, resulting in a lower movement time overall. This is shown in Figure 6.7.

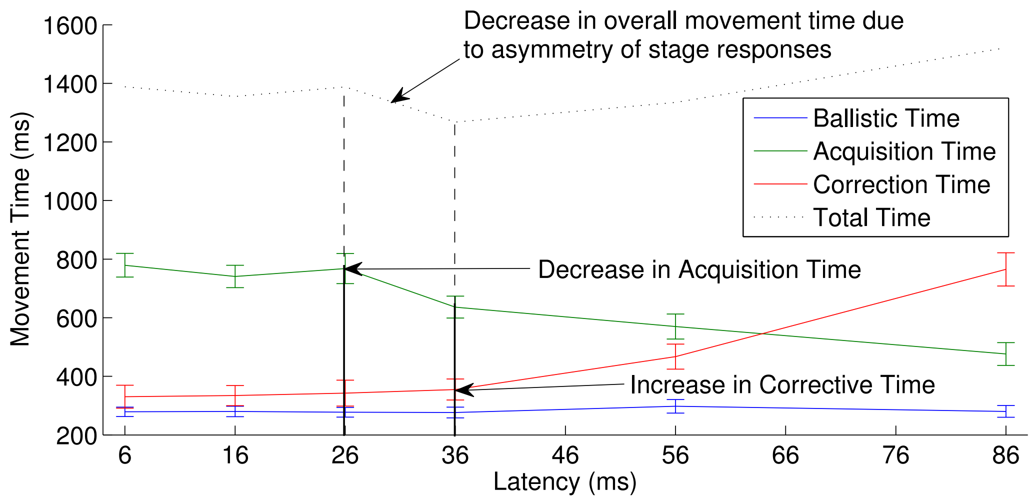


Figure 6.7: Mean time for the different stages of movement for condition 2 with latency. The graph has been annotated to illustrate how movement time changes with latency due to the differences in the response of the stages. Error bars indicate confidence intervals at 95%.

At latencies between 26-36 ms, the user does not need to make significant corrections once the target is reached, but neither does their deceleration profile match the conditions between 0-26 ms. They continue their quick movements causing them to move farther and faster than they likely intended. However, the latency is still low enough that the overshoot, if present at all, is marginal, and the correction stage is not significantly confounded. We cannot say with certainty the cause of this change in profile. One possibility is the transition to a third and unanticipated compensation process. Another is that the beginning of the deceleration is delayed due to interference with the motor processes. The result though, is that total movement time decreases with the decrease in acquisition time, until the point at which the correction stage is significantly affected, negating and then eclipsing the acquisition time gains. If this is the case, it is likely not optimal functioning of the control loop however. Interfering with the deceleration process may prevent normal trajectory modifications as the target is approached, benefiting only the subset of tasks which can be completed without these. In most cases without the controlled deceleration stage, overshoots are likely to occur and take considerable time to correct.

6.3.1.3 Latency Thresholds

With the movement stages split up we are in a better position to observe when latency begins to impact the function of the motor system. We perform ANOVA with the various pairs of latency for both movement stages. (Recall that the time in the acceleration stage is independent of latency.) Pairs between which the time in the stage differs significantly are shown in Table 6.3. For all tests $betweengroupdegreeoffreedom = 1$ and $withingroupdegreeoffreedom = 58$. From this it is clear that for these tests latency begins to have an effect at ~16 ms, ~6 ms larger than that found by Jota et al. for direct interaction.

Condition	**Latency Condition Pairs**				
	6-16	**16-26**	**26-36**	**36-56**	**56-86**
			Acquisition Stage		
1		<0.05		<0.05	
2			<0.05	<0.05	<0.05
3			<0.05	<0.05	
4			<0.05	<0.05	
			Correction Stage		
1		<0.05	<0.05	<0.05	<0.05
2				<0.05	<0.05
3			<0.05		<0.05
4				<0.05	<0.05

Table 6.3: P-Values for ANOVA between pairs of latencies within each condition

6.3.1.4 Effects of Target Width & Distance

From Figure 6.4 we see that the unexpected decrease in movement time is largest for conditions 2 & 4, which have the largest target distance. This is intuitive. The gain is a result of confounding the acquisition stage: the longer this stage lasts, the larger the effect on total movement time. From Table 6.3 we see that correction time begins to be affected at higher latencies than acquisition time. The velocity for the correction stage is lower than for the acceleration or acquisition stages. This may make the processes of this stage more tolerant to delay. Any benefits are short lived though. When latency does begin to affect this stage, the performance degradation is severe and increases rapidly. As shown in Table 6.2, while distance has a larger effect on acquisition time than correction time as would be expected, neither stage is a product only of width or distance.

6.3.1.5 Modelling the effects of latency

MacKenzie & Ware modified Fitts's law to account for the multiplicative affects of latency. We used their model, while also creating an additional linear model with predictors of width, distance and latency. The response variable is the total MT. This model is the identical to one consisting of the sum of the linear models for each of the movement stages. We consider how well both models fit our data. We selected three latencies, and for each in turn, removed the conditions with those latencies from our results. The models were fitted to the remaining data, and then used to estimate the results of omitted conditions. The error of these estimations were averaged. The results are shown in Table 6.4.

Latency Condition Predicted	**Estimation Error (ms)**	
	Our Linear Model	**MacKenzie & Ware's Fitts's law variant**
6	81.5934	78.4651
36	68.1741	68.3698
86	199.0222	201.4407

Table 6.4: Mean estimation errors of our linear model and MacKenzie & Ware's variant of Fitts's law, after estimating the movement time of a specific condition, the results of which had been removed

As the latency of the condition being predicted increases, so does the prediction error. In both cases this error is caused by an underestimation of MT. We hypothesise the response to latency is non-linear, and it if is, this is what would be expected. Since the models are created from a region of the non-linear response that that has a weaker correlation with latency, they underestimate MT as latency increases. We do not attempt to model this non-linear relationship, as we do not have data from a wide enough range of conditions to do so. Acquisition time is shown to decrease with latency, though clearly this cannot continue indefinitely. We would expect it to continue down until it reaches a floor at which it remains a constant multiple of width and distance. Since the acceleration stage is constant, and we expect the acquisition time to degenerate to constant, we must conclude that correction time will come to resemble the relationship described by MacKenzie & Ware's model. Our latency conditions are not extensive enough to test these hypotheses however so any model we created would be incomplete. This is a subject for future work. The models created if all observations are considered are shown in Equation 6.1 (MacKenzie & Ware) and 6.2 (Ours). These models have R^2 values of 0.384 and 0.685, respectively. The error degrees of freedom of our model is 716.

$$MT = 296 + (184 + 0.38LAG)ID \tag{6.1}$$

$$MT = 941 - 453Width + 45Distance + 1.6Latency \tag{6.2}$$

Two additional Fitts's law metrics that are commonly used are throughput and error. In our study, a trial was not complete until the target had been acquired. Therefore error is approximated by Correction Time. Throughput (or Bandwidth) is given as the ration between ID and MT [168]. We calculated both error and throughput and found that they had similar profiles to the MT for each condition. It is not clear how we could unambiguously separate these high level metrics into their constituent parts though, so we unable to determine any more from them than we are total MT.

6.3.2 Steering Tasks

The MT for a Steering law test is considered to be time between the cursor first touching ball, and the ball reaching the last point on the path. Like the Fitts's law tests, MT increases with ID, and there is generally a multiplicative effect of latency with ID (Figure 6.8). Although this degenerates at lower latencies (Figure 6.9). We are not aware of any previous studies that have investigated the effect of latency on the Steering law itself. The experiment closest to ours is that of Pavlovych & Stuerzlinger, in which the authors investigated the effects of latency and jitter on performance in tracking tasks [169]. As velocity was fixed in their tracking task, the performance measure was the error rate, defined as the distance from the target. Their participants performed best with 50 ms of latency (the lowest latency was 25 ms). We observe similar profiles for our conditions, in terms of MT (Figure 6.9) and error rate (Figure 6.10) though our participants performed best at slightly lower latencies.

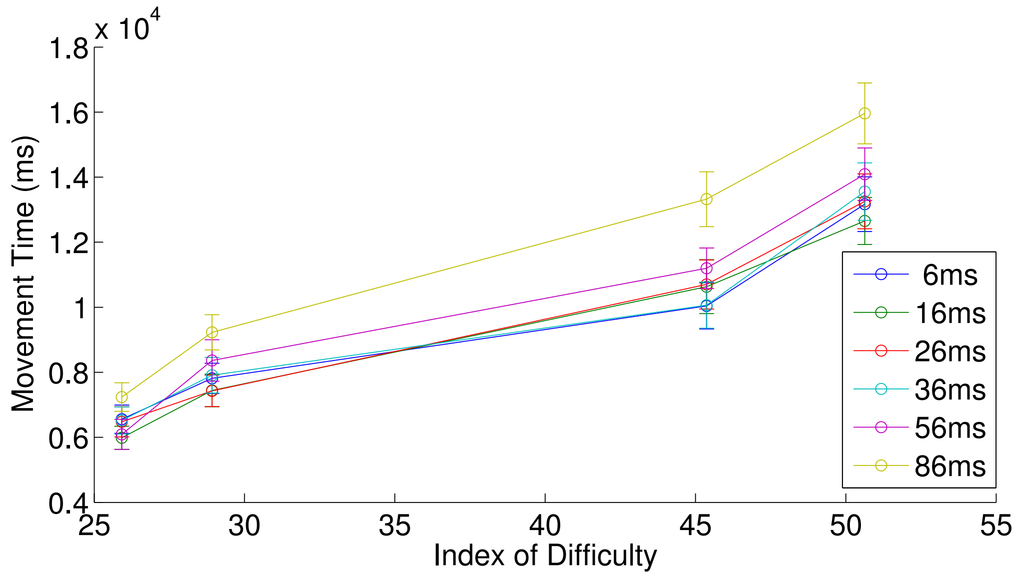


Figure 6.8: Movement Time against Index of Difficulty for all latencies. Error bars indicate confidence intervals at 95%.

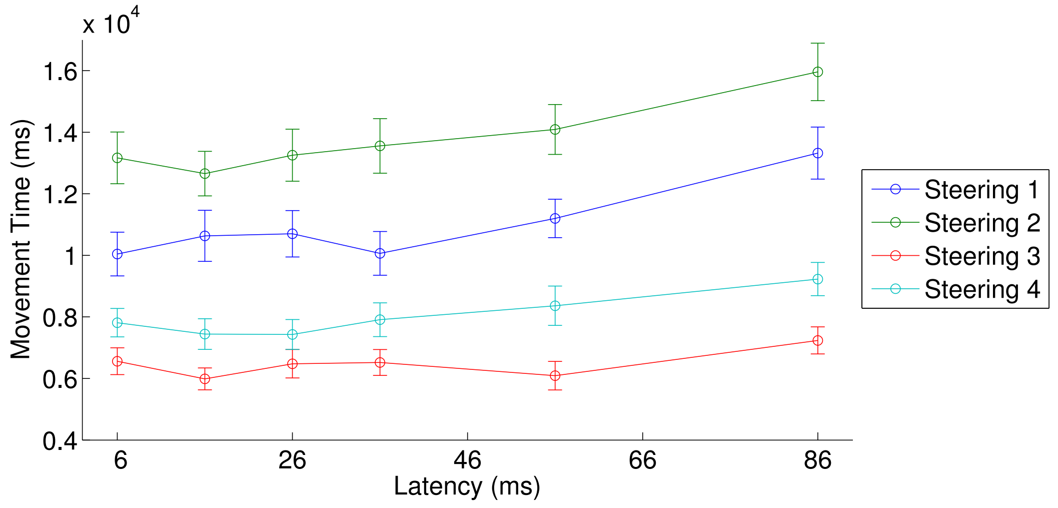


Figure 6.9: Movement Time against Latency for each Steering law Condition. Error bars indicate confidence intervals at 95%.

Path tracing is likely to require constant acceleration. Intuitively then steering tasks could be thought of as a sequence of correction movements. However, after comparing Fitts's law and the Steering law, Liu et al. postulate that behaviour is more like a series of ballistic tasks [123]. The similarity of the responses in the Steering law & Fitts's law tests in our experiments suggest both tasks use similar processes. We performed the Steering law tests as it was hypothesized they would exercise different motor processes than Fitts's law. This could help disambiguate the Fitts's law results. There is no evidence to suggest this is the case however, and segmenting movement stages in steering tasks is not as straightforward as in discrete pointing tasks. We are unable to offer an explanation for the apparent non-linear effects of latency, other than it is possibly the same interaction between motor processes seen in the discrete tests.

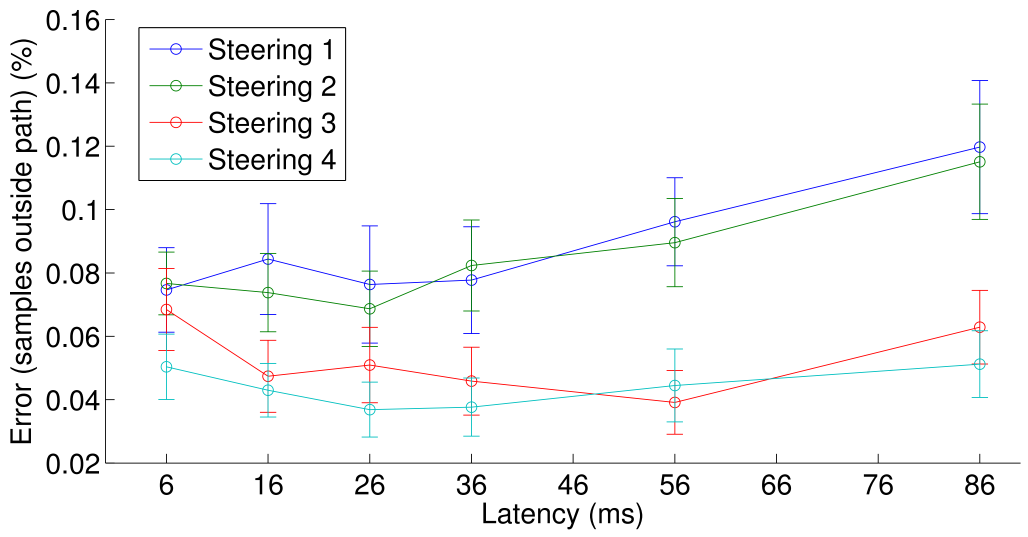


Figure 6.10: Error Rate in percentage of samples that occurred outside the path, against latency for all Steering law conditions. Error bars indicate confidence intervals at 95%.

6.4 Discussion

6.4.1 The non-linear response of latency

Even the simplest models of the motor system consider a set of separate processes, cooperating to execute smooth motion. So far though, the impact of latency has been modelled as linear. It is theorized that latency interferes with the ability to make quick corrections to motion, slowing the user down. Our results show that while this is true, it may be obscured by the interaction of latency with a stage of movement which we term the acquisition stage. When the user enters this stage they are moving at their highest velocity, and during the stage begin to decelerate. Interfering with this stage then could result in a decrease in movement time, as the user moves further and faster than they would if they had full control of their motor system. Usually this results in overshoot. With certain task difficulties and low enough latencies however, the corrections required are minimal, resulting in a lower overall movement time. It appears as if latency is improving user performance. The reality though is just another trade-off analogous to that between speed and accuracy - but one the user has no control over. This explanation also implies that the motor system is naturally conservative.

6.4.2 Thresholds of latency

Our conditions are not extensive enough to derive better models of the effects of latency on MT than the existing linear ones. By performing ANOVA for latency condition pairs though we can determine an initial range where latency does not appear to have a significant effect. The value of this threshold (16-26 ms) is not as important as the fact that such a threshold can exist. If a threshold exists for our apparatus, one may exist for more complex installations.

We hypothesize that above specific latencies, the impact of latency on the acquisition stage will degenerate. It will become constant, like the acceleration stage, and the correction stage can be modelled by the original Fitts's law with latency introduced by MacKenzie & Ware. Fitts's law describes an observation rather than the operation of the underlying system, and so it transcends the revisions of motor system models. Conversely though, it only applies to conditions within a certain range. If predicting user performance across the full range of conditions is important, models which describe the contribution of all movement processes will have to be derived.

6.4.3 Investigating Latencies at Low Levels

The Steering law tests were included in order to disambiguate the results of the Fitts's law tests, where the effects of latency on different motor processes may not be clear. In fact though, it was the Steering law test behaviour we could not explain, due to the inability to quantify participant behaviour beyond MT and error rate. The Fitts's law test is valuable in investigating latency, so long as metrics beyond that of MT are considered. It is not clear though, whether the floors we have supposedly found are functions of the motor system, or the motor system & task difficulty & apparatus. Ideally a task would be designed, which both approximated real interaction primitives, and exercised the motor processes in such a way that the response to latency remained linear, or at least linear with a clearly defined floor. That the effect of latency on the movement stages is independent is significant. If only the sum of these stages is considered, the effect of latency may be obscured. Recall that no interaction with latency was demonstrated when considering the total movement time for the lowest latency conditions. What is not clear is the extent to which each movement stage is dependent on pre-planning. In the previous tests we have considered, the requisite movements are predictable for the user. Even in Pavlovych & Stuerzlinger's tracking task, Lissajous Curves were used which made the motion of the target predictable for most of the experiment. With improved models of the motor system it may be possible to isolate and test the processes involved with each movement stage separately. Latency is detrimental to interfaces facilitating continuous or pseudo-physical interaction with complex datasets or systems. This is especially true for those affording natural interaction such as virtual environments. If we are to continue to investigate low latency, it may be worthwhile to pursue a new interaction benchmark.

6.5 Summary

Latency is known to impact performance in motor tasks. User performance has been characterized by models such as Fitts's law, which have been extended to include the effects of latency. Authors have commented previously that Fitts's law is likely to degrade at extreme values. One example is the negative intercept of the linear model, which would result in a zero or negative MT for low enough IDs. Clearly the real response must deviate from this model. We suspected a similar deviation would occur from the latency model for very low values, and designed an experiment to test this.

We constructed a system with a system latency of ~1 ms and display latency of ~5 ms. Informed by pre-trials, we selected conditions which matched those from previous works, and which happen to be those at which the motor system can theoretically operate optimally. Where conditions overlap we compare our results and find no significant differences. For our lower latency conditions however, we find a significant and unexpected effect.

Our results show that for some conditions, higher latencies can result in lower movement times. This effect has been hinted at in previous studies, though not significantly enough to pursue. We caution that movement time is just one metric and does not necessarily mean performance is improved. On closer inspection we suggest that the effect may be explained by the independent, degrading effects of latency on the processes of the motor system. While efficient user interaction is am important part of forming the sensorimotor loop, we know that the raw ‘bandwidth’ of user interaction on its own is not a sufficient indicator of the effectiveness of a synthetic environment. In the next chapter we describe a study that attempted to examine a higher level perceptual skill.

Chapter 7

The Non-Existent Effects of Latency on Distance Estimation

In Chapter 6 we reviewed a study into the effects of latency on low level interaction. Low level interaction is key to forming the sensorimotor loop, and so is very important for effective synthetic environments. Many authors have hypothesised that presence is not only a combination of various functionalities, but can be experienced at different levels. Users behave as if the world is real, but intellectually they know it is not. An example would be avoiding an obstacle or hazard they know is not there. While a considerable amount of work has been done on lower level interaction, these findings cannot extrapolate to higher level functionalities such as embodiment, perception or co-presence, yet these functionalities are just as important.

In this chapter we describe a study conducted into distance estimation. Distance estimation is an interesting skill to study because while it is higher level than pointing and reaching tasks, there is also a clear objective benchmark for performance. It is also highly important for many applications of synthetic environments.

7.1 Introduction

One persistent and widely observed phenomena in VR is distance *compression*. This is where users in a VE underestimate distance by up to 50% [32]. This has been measured using a variety of techniques, such as verbalisation [110], throwing tasks [185] or - the most popular - blind walking trials [125]. Clearly, in an application designed to facilitate or evaluate knowledge transfer or muscle memory, this is a significant problem.

Accordingly, many studies have attempted to elucidate the causes of the compression. Authors have examined factors from the mechanics of wearing a HMD [245], to the furniture in the virtual world [86]. Despite this, few have been able to find even significant predictors of the effect, let alone hypothesise an underlying cause. One of the most interesting discoveries was made by Interrante et al., who found that the distance compression disappears when the VE is a replica of the users concurrently occupied real-world location [85]. The phenomena was persistent however in follow up studies, appearing under even small changes - even when participants reported no perceptual differences [86]. Steinicke et al. showed that the benefits of a replica environment remain after exposure to it has ended. They found participants' behaviour & self-reports in a synthetic VE changed significantly when they were first exposed to a replica transitional 'ante-room' [218]. Steinicke et al. later found these benefits extended to distance estimation [217]. While the benefits have been shown to remain beyond initial exposure, the user will by necessity have experienced the real-world equivalent of the VE. The question remains then whether the improved accuracy is due to higher level cognitive functions or lower level calibrations/learning effects in the visuomotor system. So far there is little conclusive evidence for either side.

In this study we suggested that the compression may be a consequence of dynamic visuomotor processes, and that the interference with these by latency may be the cause. Latency has been shown many times to interfere with VEs at many levels, including visuomotor processes in pointing & steering tasks and presence (e.g. [129, 143]). Latency has been shown to have an effect below the consciously perceptible thresholds [91]. It is has been shown to be amenable to adaptation [23], as have distance judgements [149]. As a consequence of the unavoidable processing and transport delays in current computer systems, it will also have been present in every virtual reality experiment performed thus far.

To investigate the effects of latency, we conducted an experiment based on that of Interrante et al. [85]. We constructed a VE system with a latency of $1-4ms$, using a state-of-the-art tracking system and a bespoke graphics controller. Using this system, we asked participants to conduct a series of blind-walking trials to gauge the accuracy of their distance perception after exposure to the VE with different levels of latency. In addition, we measured a number of gait parameters, reasoning that the correlations, or lack of, between changes in latency, accuracy and gait parameters could provide important clues as to the source of the effect.

Our findings were unexpected. We found no effect of latency on accuracy or gait. However we did find that participants showed no greater compression in our synthetic VE than our replica - a VE which they could not have adapted to. The most obvious explanation is that our experiment failed in some way, however numerous tests do not bear this out, suggesting rather that that we have found a condition which facilitates accurate distance judgements in an entirely synthetic VE with no real-world counterpart.

While gait measurements could not help us refine this hypothesis further as we had hoped, this one conclusion on its own suggests it is possible to design entirely synthetic VEs which facilitate veridical distance estimation. The implication is that distance compression may be function of plausibility and believability, rather than lower level physiomotor confounds. We could not have performed this study without our high performance bespoke VE system, but now that we have, we expect our main result to be repeatable in many off-the-shelf consumer VR systems such as the HTC Vive. If this is the case, whether the intention is to inhibit or elicit distance compression, the implications for the designers of VEs and their installations are significant.

7.2 Survey of studies on distance estimation

Many applications of synthetic environments require users to perform goal directed actions in space. Accordingly, there has been interest in how users perceive & interact with VR, and what phenomena may affect such interaction. One of these phenomena is the systematic underestimation of distance [189, 32].

Users consistently underestimate distance in VR. Loomis et al. were one of the first to demonstrate this using blind walking trials [125]. Since then, authors have demonstrated the persistence of this phenomena in throwing tasks [185], verbal feedback [110], and in both speed [13] and distance [248] estimations during treadmill walking. The effect has been repeated in numerous studies which have investigated it since (e.g. [49, 59, 149, 175, 217, 225]), while authors also remark on participants moving 'slowly' or 'uncertainly' (e.g. [176, 55, 85, 187]). Despite all this attention, the underlying cause has proved elusive.

Understanding the distance compression phenomena is important, because the natural behaviour of users is arguably a key indicator, if not a pre-requisite, for an effective VE. As Slater et al. assert, presence is a phenomena enabled by the high level of immersion of VEs but distinct from simpler interaction [211]. The goal of a VE is to substitute virtual stimuli for real, and it has not successfully done this until the user forms percepts from the virtual stimuli and responds realistically to it. Meehan et al. consider that the effectiveness of a VE may be measured through the user's internal state, rather than simply application success [143]. They hypothesize that 'real' physiological responses would be evoked to the degree that a VE seemed real [142]. Usoh et al. demonstrated such an interdependence, showing the greater the degree to which locomotion techniques approximated natural movement, the higher the measures of presence [230]. Phillips et al. suggest that basic physiomotor characteristics, such as gait, could themselves be used as an objective presence measure [176, 172].

Accordingly, a number of authors have attempted to identify the factors that underlie the phenomena. Fortenbaugh et al. suggested that visual quality may cause users to utilise different cues during path integration [57]. Thompson et al. found no effect of this on distance judgements [226], however in a subsequent study Phillips et al. found that the representation of a VE does matter in some situations [174]. Kunz et al. found that graphics quality did have an effect on users' verbalised judgements, but not on their actions [110]. Campos et al. investigated FOV, but found no significant interaction [32]. Interrante et al. hypothesised that users may take cues from landmarks such as furniture, but were unable to find any evidence to support this when testing behaviour in VEs with and without virtual furniture [86]. Willemsen et al. did find a significant interaction with the mechanics of wearing an HMD, but it was not strong enough to explain the phenomena on its own [245].

Other authors have looked for an explanation in the dynamics of spatial interaction. Frenz et al. investigated the perception of travel distance from visual motion and found that distance is underestimated under motion, even with improving cues of the static layout of the scene [59]. Banton et al. found a significant interaction between the mis-perception of speed and restrictions in visible lamellar flow [13]. Campos et al. found that FOV was a significant contributor to a number of functions, but could not on its own explain the estimation compression of walked distances [33]. Thompson et al. found that by deliberately mismatching the visual cues to the users real walking speed, they could increase the accuracy of distance judgements. They demonstrated this with blind walking trials performed after exposure to a VE on a treadmill [225].

Mohler et al. also showed the plasticity of the effect. In their experiment participants performed tasks similar to blind walking trials but with different levels of feedback, before performing traditional blind walking trials to assess what effect the feedback had on their performance. The authors found that mismatching the coupling between visual and biomechanical speed in the adaption stage did transfer to the real world [149]. Rieser et al. demonstrated not only the plasticity of a number of visuomotor functionalities, but also that the re-calibration was 'functionally based', rather than action or global based [184].

One of the most interesting discoveries about this phenomena was made by Interrante et al., who found that when participants were placed into a replica of their concurrently occupied real environment, the compression effect disappeared [85]. In a follow up study [86] they found that it returns when the scale of the replica is altered, but that the mis-perception is not correlated with with the direction of scale, but merely the existence of it. The authors suggested that the phenomena could be cognitive, rather than functional. They found that participants' behaviour changed over the trials, suggesting some form of adaptation does take place, but there was no conclusive evidence that this was either functional or cognitive. Recall that Phillips et al. did find a significant interaction between rendering style and distance perception [174]. In another study Phillips et al. also found a significant effect for whether or not the participant had an avatar, though again the effect was not strong enough to completely explain the phenomena [175]. Results such as these however could suggest a strong presence component in accurate spatial judgements.

Following from Slater et al. and Meehan et al., Phillips et al. [172, 173] examined correlations between distance perception accuracy, presence and various physiological cues, including gait. In an environment designed to provoke stress as a technique to measure presence, the authors found that participants moved more ‘cautiously’ in a dangerous environment. The authors found a significant effect between locale and gait in one experiment, and locale and distance estimation in another. Again, the strongest results were found where a participant was in a virtual replica of their real world location. There was surprisingly no correlation between gait and distance estimation in the second experiment, however [173]. Steinicke et al. [218] explored the benefits of a replica environment further, examining if using one as a transitional ante-room could increase presence when participants moved into a non-replica locale. They found that it did indeed increase presence. Based on the observed change in participant behaviour, and remarks that they had a ‘better feeling for movements’, Steinicke et al. [217] pursued this to see whether the benefits extended to distance estimation. They found that participants had significantly higher accuracy after exposure to the transitional environment.

Steinicke et al. [218] coded user behaviour rather than measuring gait parameters directly. Jones et al. [97] examined the direct relationship between gait and distance estimation under different FOVs, but found that the relationship was ambiguous. Using a treadmill, Hollman et al. [77] showed that in VR, stride length was reduced, in addition to an increase in step width and stride velocity. Hollman et al. [76] further found that kinematic parameters (e.g. push-off peak force) that reflect gait stability were significantly affected. These findings suggest that participants have reduced stability. Mohler et al. [148] examined gait parameters between free walking in an HMD and in the real world. They found a significant affect on gait. Anecdotally, the authors observed that wearing the HMD began to approximate the same effects as walking 'with the eyes closed'.

So far then there is little conclusive evidence to suggest a purely functional or purely cognitive cause. Authors such as Mohler et al. and Thompson et al. used different forms of dynamic feedback to significantly influence participant behaviour in the same blind walking trials used by others to investigate 'static' cues. In doing so they provide compelling evidence for an interdependence between spatial perception and spatial dynamics [149][225]. Therefore, we cannot ignore any potential factors, even if they may appear to affect only exclusively dynamic or static cues. Specifically, we suggest that latency may be a factor. Latency has been proven to affect visuomotor processes (e.g. [61]) and presence (e.g. [143]). Latency is amenable to adaptation in some functionalities (e.g. [23]), and as the unavoidable consequence of processing and transport delays in the computer systems that make up current VEs [147], it will have been present in every study referenced above.

The authors are aware of only one group that has examined the effects of latency on gait or distance estimation. Samaraweera et al. [187] investigated the effect of both latency and avatars on the gait of participants with and without mobility impairments. They found that while only two participants consciously perceived the latency, there was a significant effort on behalf of both populations to take a more cautious approach to walking with increasing latency. In another study Samaraweera et al. [186] found that latency could significantly influence the gait of participants when it was applied asymmetrically, to only one side of an avatar seen in a self-facing mirror.

7.3 Experiment

We designed an experiment to test the effects of latency on distance estimation in VR, while at the same time elucidating how accuracy correlates with physiomotor behaviour. The experiment had two independent variables, within-subjects latency and between-subjects room. Two virtual rooms were prepared, one replica to inhibit compression and the other, entirely synthetic, which we presumed would elicit it. We continued the practice of using blind walking trials as an indicator of distance estimation in VR, following the prototype of that used by, for example, Interrante et al. [85]. After orientation, participants entered VR and alternated between free walking periods and blind walking trials. During the free walking periods, participants were exposed to different levels of additional latency: 0, 10 & 25ms. Participants experienced each latency level for four consecutive trials. Participants' head and feet were tracked, allowing for estimation accuracy and a number of gait parameters to be measured under the different conditions. By observing how participants' accuracy changed with latency, we could determine if latency was significantly affecting distance estimation. By examining how gait parameters changed with accuracy, with and without compression, we could determine to what extent compression is correlated with the behaviour of the physiomotor system.

In accordance with previous works, we expected to find a distance underestimation of up to 50% in our synthetic environment, compared to our replica environment. We expected that latency would affect gait, increasing step width, while decreasing stride length and speed. We expected similar effects in the blind trial stage (vs. visble). For each participant we recorded Gender, Age as well as level of previous exposure to HMDs and 3D video games. We did not expect Age, HMD or game exposure to affect distance estimation or gait. Studies have shown that despite common perception, the effect of gender on gait remains ambiguous. However there are suggestions men take wider steps than women, possibly due to structural differences [60]. We therefore expected gender to be significant, but with few expectations of effect size or direction. We expected, as our main hypothesis, that latency would be positively correlated with distance underestimation in the synthetic room.

7.3.1 Apparatus

The design of this experiment made strenuous demands on the VE. We integrated our real-time ray caster from Chapter 4 to form the foundation of our apparatus.

7.3.1.1 Headset

For the display we used an Oculus Rift DK2 HMD. This headset has a FOV of ~100 degrees and a resolution of 960×1080 in each eye. It features a 6 DOF IMU. The DK2 was used because of its amenability to customisation for integration with our bespoke system. It can be driven by any VESA compliant 1920x1080 HDMI signal. Full source for version 0.4.4 of the SDK is available, and it was straightforward to attach mounts for the tracker markers and their driver unit. The SDK uses USB libraries that are not available on our target system - CentOS 6.7 - so we modified it to use a different USB library. The SDK and headset were otherwise unmodified. The HMD features a low persistence display that scans out left-to-right at 75 Hz in a rolling band. Consequently, the display will take up to 12 ms to completely redraw, but the user never actually sees more than a couple of hundred lines of old frame data at any time as lines drawn previous to these are not illuminated.

7.3.1.2 Tracking

For tracking we use a PhaseSpace Impulse X2E. This is an outside-in tracking system that uses active markers. The system uses line-scan cameras which allow it to achieve a very high update rate (960Hz), and a very low latency (3 ms). The markers are LEDs driven by a micro-controller, and using a time-based flashing code, can unambiguously identify themselves. We use one marker each for the left and right foot, recording absolute position. The HMD was outfitted with 6 markers to form a rigid-body which supplied both absolute position and orientation data.

We found that that there was noticeable jitter in the rigid body orientation estimation (though less in position) near the bounds of the tracking volume. We therefore designed a very simple fusion algorithm to use the HMD's on-board IMU to track the orientation of the headset. At start-up the algorithm applies the orientation estimate from the PhaseSpace to the egocentric estimate of the IMU as a fixed offset. When the system is running, the adjusted estimate of the IMU is compared with the allocentric estimate from the PhaseSpace. The difference is low-passed with a very aggressive moving-average filter, inverted and applied to correct for drift.

While ideally we would minimise complexity by having only one tracking system, there are advantages to this configuration in addition to reducing jitter: (1) the IMU, running at 1 KHz, reduces latency below even 3 ms and (2) the orientation tracking will work even if the user reaches the edge of the tracking volume, reducing the severity of the symptoms of loss of tracking[1].

7.3.1.3 Audio

The ambient noise in the lab was quite sympathetic to our virtual worlds. The investigator remained in the same room as the participant, but communicated via a set of headphones. This was to avoid the sensation of a disembodied voice emanating from one particular place.

7.3.1.4 Software

With the exception of the bespoke graphics controller, the VE ran on a standard mid-range CentOS 6.7 PC. As the ray-caster implementation runs independently of the CPU, the actual processing requirements are quite low, as described in Section 4.4. The computer running the VE ran headless, with the VE being controlled remotely over the network through a second, lower priority thread. To avoid ever blocking the main thread, synchronisation was performed using boolean flags. These indicate to the main thread when a state change was desired (e.g. blank the display), and the main thread would routinely poll these and execute the requests. More complicated requests such as downloading the logs would stall the main thread, but these were completed only when the participant was finished in the VE.

7.3.1.5 Virtual Worlds

The nature of our graphics controller severely constrains what we can display. Our current implementation supports a small number of texture & transparency mapped planes. The procedure required one of the rooms to be a replica of the lab. We created two cuboid rooms made of six planes. All planes were textured with maps created from photographs.

Replica Lab The maps of the lab were modified to remove any furniture or features large enough that the baked perspective cues would be particularly noticeable as

[1]But does not eliminate them - positional tracking is still lost.

participants moved around the room. In the replica lab, we added a virtual curtain blocking the end of the room as this area was beyond the stable tracking volume. The visible area of the lab had a floor space of $5m \times 4.5m$. Some of this space was outside the tracking volume or otherwise occupied. By carefully choosing the marker locations and instructions given to the participant, we prevented them walking into these areas. Photos of the real lab and renders of its virtual counterpart are shown in Figure 7.1.

Synthetic Room The maps of the synthetic location were created from an amalgamation of images from various locations in our office, so that it could plausibly be a real place, but one no participant could ever have visited. The virtual room was smaller than the lab, with a visible floorspace of $3.5m \times 3m$. The room was positioned in the real lab such that its entire area was navigable. Renders of the virtual room are shown in Figure 7.2. Both this render and that of the virtual lab were done with an unlit shader that simply passed through the texture sample - the same operation performed by the realtime ray-caster.

7.3.1.6 Latency

As described in Section 5.3.2, what is perceived by a user is a function of the interaction between the rendering technique and display technology, and so latency cannot be characterised by a single value. This is especially true in cases where tracking systems run at different rates & latencies. We can characterise our tracking systems, depending on the degree of freedom, by a latency of between 1 and 3 ms. We can prove our renderer has a response time of less than 1 ms, and that the display shows pixels with a maximum age of ~1-2 ms - a function of the width of the rolling band and frame period[2]. If we had to specify a single value however, the average response time of the predominant vection cues is 2-3 ms.

7.3.2 Participants

31 participants ($\overline{Age} = 29 \pm 9, 14F, 17M$) were recruited via an advert in a participant pool and paid £10 each. 16 experienced the replica room and 15 the synthetic room.

[2]This is based on 1 ms exposure capture of the rolling band. The actual period may be less. An accurate measurement could be performed with a photodiode and an oscilloscope.

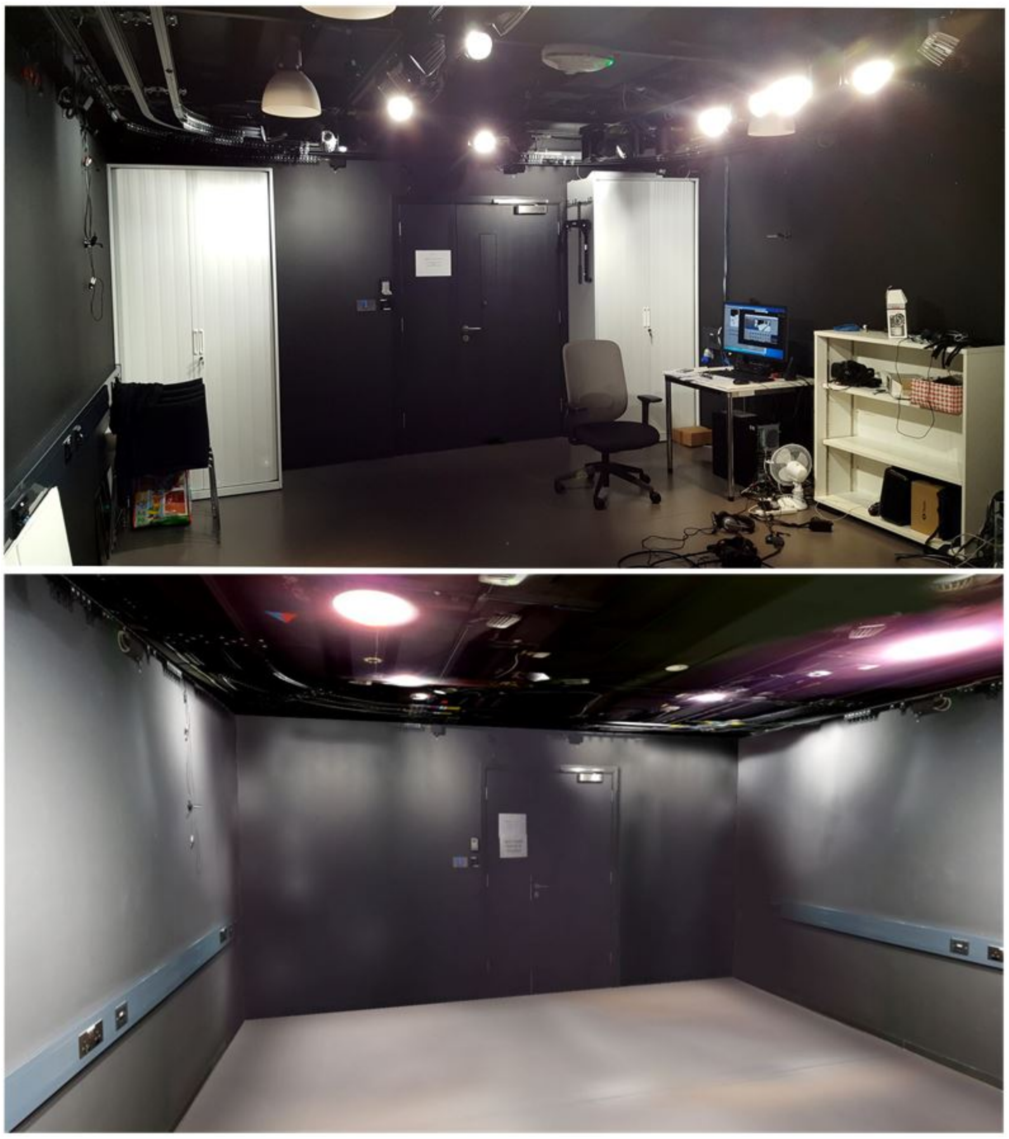


Figure 7.1: Images of the real (top - photo) and virtual (bottom - render) lab environment

7.3.3 Procedure

Participants were given an information sheet which outlined the experiment. They were told it was a study on spatial perception, but latency was not mentioned. Participants completed one practice trial outside the VE to ensure they understood the task. Participants were then asked if they were happy to proceed, and if so asked to sign a consent form. Velcro straps were attached just below the elbows, knees and around the feet. Velcro backed active markers were attached to the straps on the arms and feet, and cable slack attached to those on the knees. The markers on the arms were a diversion and not recorded. The HMD, with the virtual world already

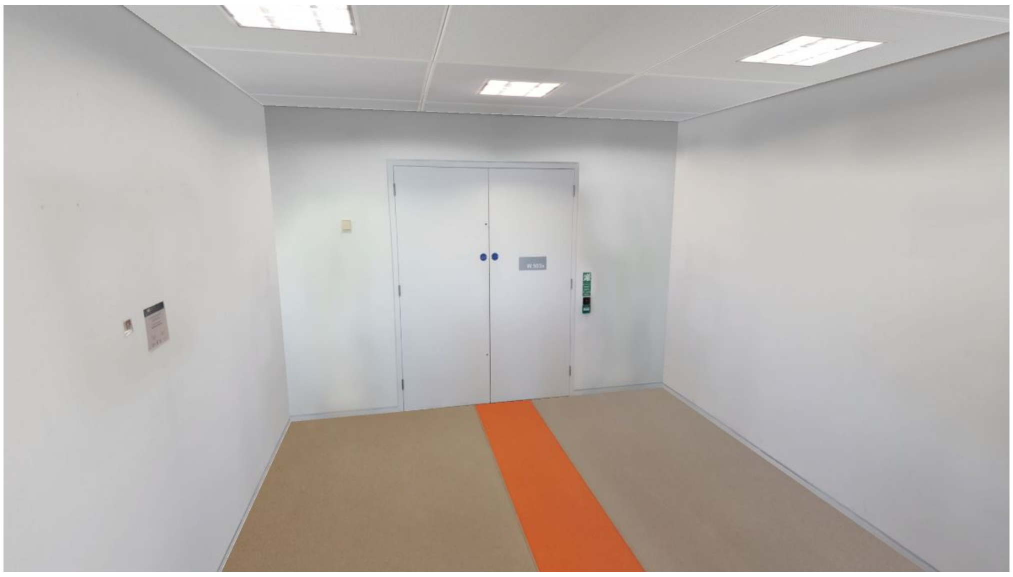


Figure 7.2: Render of the synthetic environment

loaded was then provided to the participant.

With the VE visible, but the marker invisible, the investigator asked the participant to move around the room a couple of times. Examples of instructions would be “please walk into the corner in front of you” or “please walk half way along the wall on your left”. The investigator would choose instructions so that the participants would end up on the opposite side of the room from where the marker was to appear. The investigator would then cause the marker to be shown in the HMD and ask if the participant was ready to walk. When the participant answered in the affirmative, the HMD was blanked, and the participant instructed to walk to the marker. Once they had indicated that they had done so, they were asked to take a few steps back before the VE was shown again. This was to prevent them using environmental cues as feedback about their accuracy. The procedure was then repeated for the remaining trials.

Marker positions were precomputed ahead of time for each participant, by selecting them at random from a list of potential locations. Every participant experienced all three latency conditions. Four trials were completed for each latency, and these were completed together in a block, though the blocks were ordered randomly for each participant. In total each participant completed 12 trials which took them approximately 15-20 minutes. We strictly constrained the number of conditions and repeats to limit the amount of time in the HMD, to avoid adaptation and the higher latencies inducing simulator sickness.

7.4 Analysis

We used four primary response variables which are described below. Our two main independent variables were latency and room, though four additional predictors were provided by the questionnaire results, and the actual distance to the marker could be computed for each blind-walking trial. Our experimental software automatically segmented the logs based on trial and stage (blind or visible).

7.4.1 Accuracy

Our primary measure is accuracy. We compute this as a percentage/normalised mis-estimation the same way as Interrante et al. [85]:

$$Accuracy = (WalkedDistance - TargetDistance)/TargetDistance$$

$WalkedDistance$ & $TargetDistance$ being the euclidean distance between the first head-position and last head-position & first head-position and target position, respectively. Accuracy was computed for each trial resulting in 372 data points across both rooms.

7.4.2 Gait

Previous works have examined various gait parameters. Phillips et al. [176] used Stride Speed (distance between footfalls divided by the time between them), Stride Length and Stride Width. Mohler et al. [148] used Speed (trunk velocity), Stride Length, and Head Trunk Angle. Jones et al. [97] used a variation of Step Length ($walkeddistance/numberofstepstaken$), and Speed. Hollman et al. [77] used Stride Length, Step Width and the variability of each, while Samaraweera et al. [187] observed a considerable number of parameters and found the most significant were Speed, Step Length and Stride Length. Based on the most significant and popular of these, we opt to examine Speed, Step Length and Stride Width, defined below.

7.4.2.1 Velocity

Similar to Mohler et al. [148] we attempt to measure speed as the average velocity of the trunk while the participant is neither accelerating or decelerating. We measure speed by segmenting each head position log into blocks of 500 ms, and compute the speed based on the first and last positions in these blocks. Block samples below $10cm/s$ are discarded. The remaining samples are averaged to provide a speed for that trial/stage. Speeds are computed separately for both visible and blind walking stages in each trial, resulting in 744 data points across both rooms.

7.4.2.2 Step Width

We consider Step Width to be the distance between a stationary foot and the perpendicular line made by the other, based on the definition of Baker [12].

7.4.2.3 Stride Length

We consider Stride Length to be the distance between two successive footfalls of the same foot, based on the definition of Baker [12].

7.4.2.4 Measuring Gait

Two characteristics of our experiment made it difficult to define an algorithm for automatic gait measurement: (1) We only used one tracker per foot, and that was sometimes occluded. (2) The user often changed direction (in the visible walking stage). Instead, we used manual annotation to measure gait parameters. This has the risk of introducing small errors into the measurements, but not severe outliers as the undetected failure of an automated algorithm would. A tool was created in Matlab allowing the investigators to mark footfalls or closest points along a path (an example is shown in Figure 7.3. Footfalls were identified based on the density of sample points (the sample rate was constant, so the density is an indicator of motion). The investigators annotated as many steps & strides as could be discerned, for both the visible and blind stages of each trial. Unlike speed these samples were not averaged but kept as a set of arbitrary length for each condition/stage. A total of 2469 step widths and 2269 stride lengths were annotated. Annotations were done by one investigator over two days with no indication of which trial, stage or participant the logs came from.

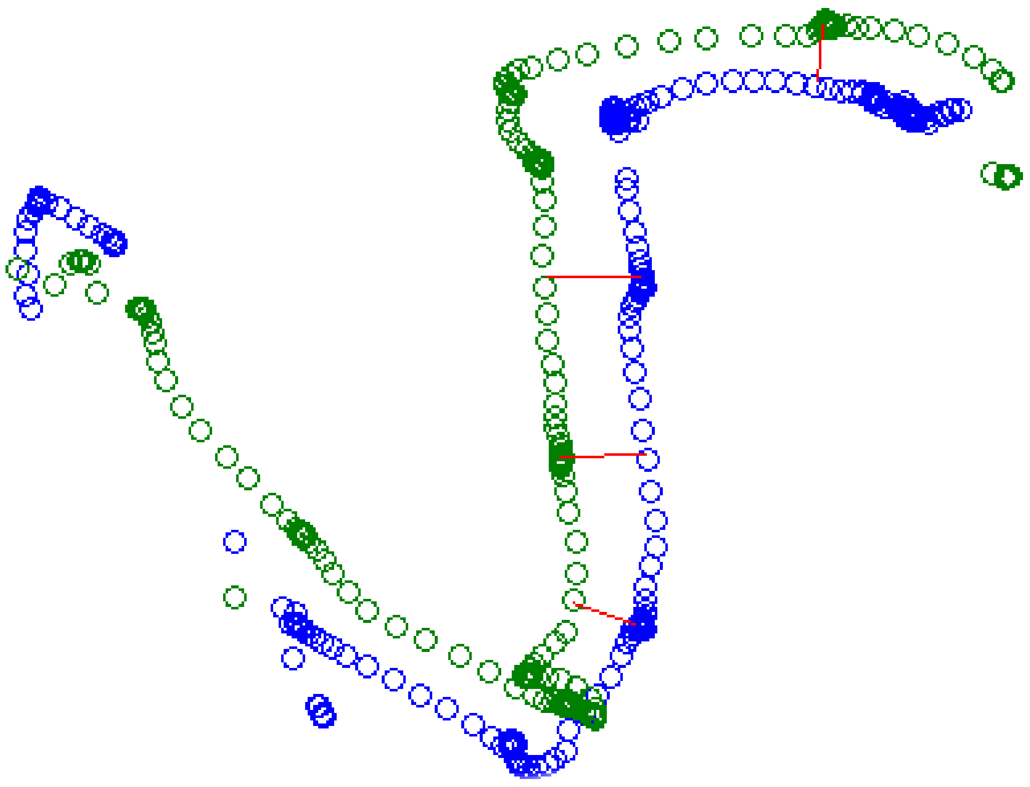


Figure 7.3: Footfall capture annotated for Step Width. The number of visible samples has been reduced to better expose the locations of highest density indicating footsteps. The annotator clicks on the image at the start & end of each red line to define the width for a step.

7.5 Results

We performed a number of statistical tests of increasing power and complexity. We examined our main hypothesis first, and then proceeded to examine interaction with gait and the effects of other predictors. Due to the unexpected nature of our results, we dedicate more effort than planned to verification and comparison with previous works.

7.5.1 Effects of Latency on Accuracy

Unlike previous studies, we had multiple repeats per condition, but not enough to do a per-participant ANOVA. We therefore perform a Friedman's Test. Friedman's

Test is similar to a one-way ANOVA but is robust to nuisance effects (such as per-participant bias). We ran a Friedman's Test for the Accuracy measure, grouped by Latency. We found no effect of Latency ($p > 0.05$) in either the Replica room $\chi^2(2,30) = 2.49$ or Synthetic room $\chi^2(2,28) = 4.02$.

7.5.2 Effects of Latency & Room on Accuracy & Gait

We next tested for the influence of gait, performing a mixed-design 2x3 ANOVA. We performed this test for the measures Accuracy, Speed, Step Width and Stride Length. The three within-subjects factors were the three latency conditions and the between-subjects factor was the room. Repetitions for each participant were averaged for each condition (only the visible walking stage for gait parameters was examined), resulting in a single sample per-participant per-cell for all measures.

For Accuracy there was no interaction ($p > 0.05$) with latency $F(2,58) = 0.097$ or room $F(2,58) = 2.17$. For Speed there was no interaction ($p > 0.05$) with latency $F(2,58) = 0.59$ or room $F(2,58) = 0.75$. For Step Width there was a significant effect ($p < 0.01$) of latency $F(2,58) = 5.12$ but not with room $F(2,58) = 1.08$. For Stride Length there was no interaction ($p > 0.05$) with latency $F(2,58) = 0.82$ nor with room $F(2,58) = 2.61$.

Samples were normally distributed within each cell, except for the *Synthetic Room, 10 ms* condition which had a slight skew.

7.5.3 Mixed Linear Models for Accuracy & Gait

While ANOVA is a convenient test, it is not the most suitable for our experiment. It is limited in power given the need to average out multiple samples per participant. Further it would soon become unwieldy were we to try and examine additional factors such as the questionnaire responses. We therefore perform a multilevel analysis for each measure. The multilevel analysis (or mixed-effects model) is similar to a linear regression but with support for random-effects - per-group coefficients which can control for group level effects, such as individual biases in a repeated measures design.

Readers should note that we are not trying to build a model of the effects of latency. Not only is our data insufficient, but our previous work suggests that this may be beyond a simple linear model regardless [61]. The mixed-effects model is a convenient way to test the significance of a number potential factors.

In a mixed-linear model the significance of the random effects can be tested by performing a Likelihood Ratio Test, comparing a model with the predictor to one without and determining if it is a significant improvement. We do this for all group level predictors. Given the large number of interactions we are testing, we consider only highly significant interactions ($p < 0.01$).

7.5.3.1 Accuracy

We defined a mixed-effects model for latency as:

$$\begin{aligned} Accuracy = 1 + Latency + Age + Trial \\ + (1|HMD) + (1|Room) + (1|Gender) + (1|Games) \\ + (1|Participant) + (Latency - 1|Participant) \end{aligned} \tag{7.1}$$

That is, Latency, Age and Trial ID were considered fixed effects (constant across subjects) while Room, Gender, Participant ID and previous exposure to HMDs or Games were considered random effects on the intercept (creating an individual bias independent of Latency). We also include a per-participant effect of Latency.

The Accuracy model showed none of the fixed or random effects were significant predictors, except the per-participant intercept $\chi^2(1) = 52.13, p < 0.01$. Again our intent is not to create a model, but for completeness $R^2 = 0.48$.

7.5.3.2 Speed

We defined the same mixed-effects model for Speed, Step Width and Stride Length, which was the same as that for Accuracy but with an additional Visibility term to distinguish between the visible and blind walking stages of each trial.

$$\begin{aligned} Measure = 1 + Latency + Age + Trial \\ + (1|Visibility) + (1|HMD) + (1|Room) + (1|Gender) + (1|Games) \\ + (1|Participant) + (Latency - 1|Participant) \quad (7.2) \end{aligned}$$

When applying this model to Speed we found as expected a significant per-participant intercept $\chi^2(1) = 158.34, p = 0$. We also found that Visibility was significant $\chi^2(1) = 193.34, p = 0$. The direction of the BLUP (Best Linear Unbiased Predictor) coefficient was unexpected however ($\beta_{yes} = 3.18cm/s, \beta_{no} = -3.18cm/s$) (a change of 15% of the average) suggesting that participants walk faster in the blind stage. This is contrary to expectations that users would walk more cautiously in more uncertain conditions. While we cannot say for certain, we highly suspect this is a consequence of our experimental design. Within the visible stages the participants walked typically shorter paths in short bursts, whereas in the blind walking stages they made one movement, typically over a much larger distance allowing them to build up a rhythm. It is not possible to prove this, but we do show a significant correlation with distance ($Spearman : \rho = 0.27, p < 0.01$), ($Pearson : r = 0.25, p < 0.01$). Latency, Age, Trial, Room, Gender, HMD & Games were not significant ($R^2 = 0.55$).

7.5.3.3 Step Width

When applying the model to Step Width, we found again the expected significant per-participant intercept ($\chi^2(1) = 132.35, p = 0$), and also an effect of gender in the expected direction ($\chi^2(1) = 11.63, p < 0.01, \beta_M = 1.9cm, \beta_F = -1.9cm$). We found that the Visibility predictor was significant ($\chi^2(1) = 9.29, p < 0.01$), but meaningless, with the BLUP ($\pm 0.27cm$) less than both the standard error and inconsequential compared to the mean ($\sim 23cm$) or even effect of gender ($\sim 4cm$). The fixed effect Trial ID is also significant in the expected direction ($t(2464) = -3.60, p < 0.01, \beta = -0.10cm$), but the effect size ($< 1\% mean$) renders it meaningless. Participants spent approximately the same amount of time in each trial, so ID, which increases from $1 - 12$ could be considered continuous and a good approximation of time. Latency, Age, Room, HMD & Games were not significant. $R^2 = 0.36$.

7.5.3.4 Stride Length

When applying the model to Stride Length the usual per-participant intercept is significant ($\chi^2(1) = 325.36, p = 0$). Also significant are both Trial ID ($t(2265) = 9.35, p < 0.01, \beta = 1.25$) and Visibility ($\chi^2 = 13.72, p < 0.01, \beta_{yes} = -1.63cm, \beta_{no} = 1.63cm, SE = 1.72cm$)), though again the coefficients (both $1-2\%$) are so small as to be meaningless. As with Speed, there is a smaller though still significant correlation with distance ($Spearman : \rho = 0.10, p < 0.01$), ($Pearson : r = 0.08, p < 0.01$). Interestingly, the random (per-group) effect of latency is significant $\chi^2(1) = 28.19, p < 0.01$. Latency, Age, Room, Gender, HMD & Games were not significant ($R^2 = 0.38$).

7.5.4 Comparison with Previous Works

Before attempting to draw any conclusions we verify that our measures are consistent with previous studies. The most similar studies to ours which reported gait parameter values were those of Mohler et al. [148], Phillips et al. [173] & Samaraweera et al. [187]. A summary of overlapping conditions is shown in Table 7.1. As can be seen, the most significant deviation is in Speed, for which we recorded much slower values than other studies. Otherwise our measures are within intra-study variance, and almost indistinguishable from Phillips et al.

Other studies that were similar but not directly comparable due to protocol differences include those of Jones et al. [97] and Hollman et al. [77], who recorded Stride Lengths, or measures analogous to stride length, of 100-130cm (similar to those in Table 7.1).

7.5.5 Replica vs. Synthetic Environments

Our results above are consistent, at worst we can say that factors have ambiguous significance. We have performed a number of statistical tests and none of them can find any indication of an effect of latency on distance estimation. While we had hoped to find an effect, the lack of one should not be that surprising, given the

Measure	Condition	Mohler et al.		Phillips et al.		Samaraweera et al.		Ours	
		Mean	Std	Mean	Std	Mean	Std	Mean	Std
Speed (cm/s)	Visible Replica VE			76.00	15.00			36.75	6.09
	Visible Synthetic VE	126.00	8.00	61.00	20.00	53.00	12.00	37.98	6.84
	Blind Replica VE							44.57	9.55
	Blind Synthetic VE	107.00	5.00					42.85	8.50
Stride Length (cm)	Visible Replica VE			94.90	12.53			93.71	24.52
	Visible Synthetic VE	128.00	6.70	84.00	20.20			86.99	23.76
	Blind Replica VE							94.84	25.49
	Blind Synthetic VE	105.00	3.80					88.40	20.84
Stride Width (cm)	Visible Replica VE			21.10	2.51			23.71	5.68
	Visible Synthetic VE			22.33	3.92			24.70	4.78
	Blind Replica VE							22.88	5.05
	Blind Synthetic VE							24.03	4.70

Table 7.1: Absolute measures of gait for the closest conditions to ours from previous works [148][173][187]

number of potential factors that have been investigated previously.

What is surprising however is that there is no difference in accuracy between the Replica VE and Synthetic VE. None of the mixed-effects models showed the Room factor as significant. For completeness we ran a Wilcoxian Rank-Sum Test (when all latencies are concatenated the distribution is no longer normal) between the Rooms and found no significant difference $(Z = 1.38, p = 0.16)$.

Interrante et al. has shown that the underestimation disappears when the VE is a replica of the participant's real-world location, leading us to expect a significant difference between our two environments [85, 86]. The obvious explanation would be our Replica environment is not high fidelity enough - except that the underestimation in both rooms was consistently *low*. The mean estimation error is 7.6% with a variance of 0.015. This error is comparable with that observed by Phillips et al's replica environment (5.82%). They found that this was not significantly different from the real world error (0.11%) [173]. Interrante et al. also found errors of ~8% for both their real and replica environments [85]. Such errors are far lower than the typical VE error (e.g. 17.72% [173], $16.67-22.29\%$ [110] & 26% [59]. The estimation error is plotted for both rooms in Figure 7.4 and illustrates quite clearly that even if there is some slight effect our tests are not powerful enough to detect, it is inconsequential and significantly different from previous studies.

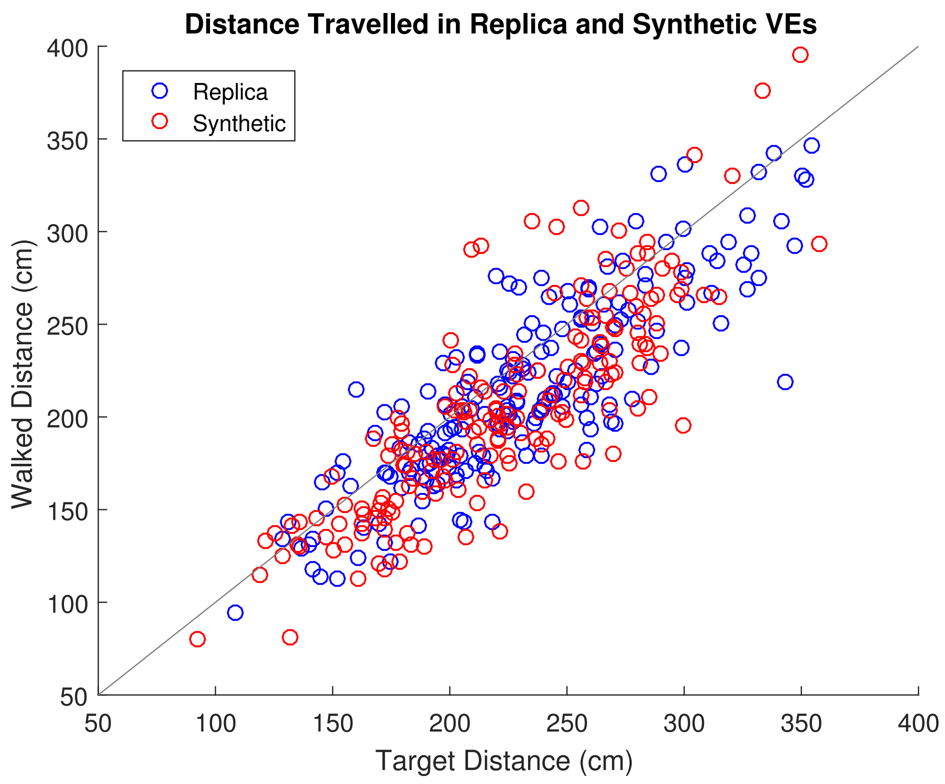


Figure 7.4: Plot of the actual distance traversed compared to the actual distance to target, for all trials of all participants

While our results were not as conclusive as we would have liked, there is no reason to believe they are erroneous. We therefore examine if the lack of effect is due to protocol differences. We modelled our experiment on those of, e.g. Interrante et al. [85] & Phillips et al. [173], so for the most part the protocols were similar, with two exceptions.

7.5.5.1 Distance

The apparent lack of underestimation may be consequence of the length and variation in the trial distances. Due to space constraints the typical distance walked by our participants is much lower than in previous studies ($0.1 - 3.5m$ (ours) vs $2.4 - 4.8m$ for Phillips et al. [173]). The most convincing evidence for this is the estimation errors reported by Kunz et al. [110] which appear to show a decrease with distance. If we extrapolate from these using a linear model[3], we find that at our mean distance of $2.09m$ the expected error is 13%. Williams et al. [246] examined distances including

[3] $Error = -0.053 + 0.089 * Distance$

those closer to our range. They found that between $2.5 - 3.5m$ the estimation error tends towards 0. However, the relationship between error and distance is linear, and so below this the error turns into an increasing overshoot, rather than disappears.

The absolute error of our participants with distance is shown in Figure 7.5. There is a significant positive correlation between absolute error and distance for our data ($\rho = 0.34, p < 0.01$), though it is not as strong as that of Williams et al.

Importantly, we can see that the error increases past the intercept. That many of our samples fall within the range of which they would tend towards zero anyway is unfortunate. It reduces the power of our statistical tests. It does not though explain the lack of an effect between the two rooms. While distance is correlated with error, it is not a case of there being a threshold, in general or in our experiment.

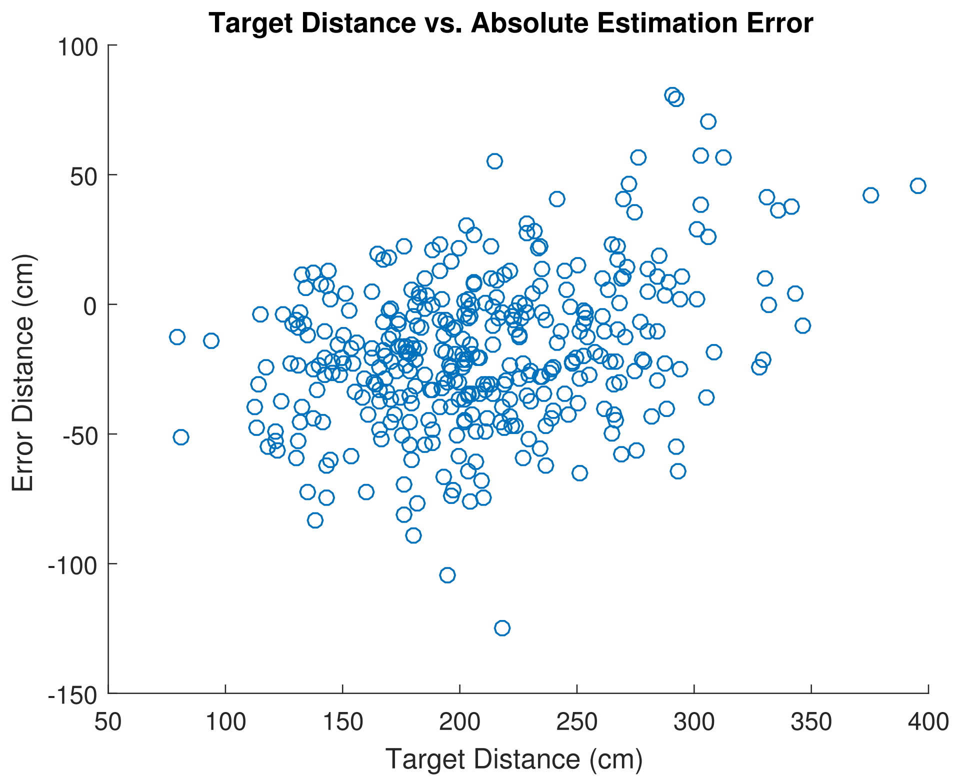


Figure 7.5: Absolute estimation error compared to initial target distance

7.5.5.2 Adaptation

The number of repeats per-participant vary between previous studies. We minimised the number of repeats to prevent nuisance effects of adaptation. While there is some suggestion from the mixed-effects models of adaptation of gait, it is weak, and there is no such suggestion for estimation error.

7.5.5.3 Apparatus

It is difficult to say if our VE system itself is significant. In some ways it is higher fidelity than those used in previous studies. For example, our DK2 has a higher FOV than Phillips et al.'s or Interrante et al.'s nVisor SX. However it is slightly below current consumer state-of-the-art in terms of FOV and resolution. Further using an older SDK with our own image distortion algorithms there was more geometric distortion in our headset than likely in other systems. Our tracking and rendering systems are state-of-the-art. Though we have dismissed the effects of latency in the range 3-28 ms, some previous works will have had latencies exceeding our highest level. The screen is driven differently to typical GPU-based systems, as described in Section 5.3.2. Unlike current commercial systems, we did not use any prediction or predictive warping and the only distortion users experienced in this sense was the 'true' latency. It is not clear if previous investigators used prediction. The latency in our system is also deterministic, unlike GPU based systems for which it will vary based on the complexity of the visible scene. Considering the content of the scene, no other authors had our limitations on geometry, so any previous study that used photos as texture maps will have had an equivalent or higher fidelity environment.

7.6 Discussion

Our original objective was to examine the effects of latency on distance estimation and gait. Interrante et al. have shown it is possible to build a VE which inhibits distance compression, and Steinicke et al. showed that the benefits of exposure to this can extend to non-replica locales. The nature of this VE - a room identical to the co-located real world - however leaves the question whether the lack of compression is due to a higher-level place illusion or 'presence' or some lower-level calibration or learning effect. Our experiment was designed to test for the effects of latency on compression, and the physiomotor system (signified by gait parameters) under conditions in which we expected compression to occur, and in which we would expect it not to. The purpose was to test if latency is an explanatory factor for distance compression, and add to the discussion on at what level the source of the compression occurs. We have in some way achieved this, if it not in the way expected.

We have not found an effect of latency on distance estimation. While this was not entirely unexpected, to great surprise we found no compression in the condition designed specifically to elicit it. This is significant because unlike previous cases, our room was entirely synthetic, meaning there was no way for the user to learn or calibrate to the room before the trials began.

This finding is interesting, but also a serious complication. We included the condition precisely because it would guarantee an example of compression. That we are unable to show any compression in our experiment raises the question - is there no effect, simply because the experiment has failed? Certainly our measures have a higher variance than Interrante et al.'s original study [85]. The range though does not differ in magnitude from subsequent ones (e.g. [174],[86]). Ultimately this hypothesis can be quickly dismissed with a basic correlation test between the target and walked distance (showing a significant effect). Our results are noisy but they appear valid.

This leaves the question, is the noise so great it is obscuring the effect? (A Type II Error). This is harder to address, as it is no more feasible to negatively prove an effect of room type, that is an effect of latency. None of our tests have shown a difference of even borderline significance between the rooms. This question however obscures the more important consideration of effect size. Even a cursory glance at Figure 7.4 will reveal that if there was an effect, the difference between the two populations is negligible, and quite different than that observed in previous studies. As to whether the noise may hide the typical VE underestimation in its entirety for both rooms, this is the same as asking, if the true population mean was 20-30% what is the probability of our sample having a mean of 7%. An Independent Two-Sample T-Test shows this to be < 0.05.

This in itself contributes to the discussion on the source of distance compression.

As there is no way for a low-level calibration to have been performed, it is relatively strong evidence for the 'high-level' explanation. Our synthetic room while having entirely novel geometry, was made up of detailed photos taken of the environment through which the participants passed to reach the lab. The synthetic room included architectural details such as door furniture, carpets and even had similar lighting.

The high vs. low level explanation could be further elucidated by the effect, or lack of, on gait, of both latency and room. Accordingly it is regrettable our gait measurements could not contribute to this. In our analyses, there were hints of both adaptation and the effects of latency, but they were very slight. Recall that our intention was not to create a model, precisely because we suspect a linear model is an insufficient characterisation of these complex interactions. It is therefore possible the small effect size may be indicative of a more important effect, but we are unable to make such a claim and can only suggest this as worthwhile future work.

It is possible these ambiguities are due to noise in the gait measures, of which there are two primary sources. The most significant is the need to manually annotate the logs. This is a consequence of tracker drop-outs for the feet and the relatively unpredictable manoeuvring of participants, both of which make automated algorithms unreliable. The first is a consequence of our tracking system configuration, and the second our limited space. If we were to re-run the experiment we would increase the tracking volume near the floor and reduce drop-outs by angling the cameras down, relying on the IMUs in the headset to provide primary tracking for the head, corrected with a second lower quality tracking system installed in our lab. We would update our protocol so that the participant trajectories were much more consistent and predictable. One way would be to actually have the participant move between a set of visible markers. Finally, given the ambiguity in our measures it may be worth including more parameters as per Samaraweera et al. [187]. Even if they did not find all of them significant, other experiments may. The weak but expected effects of gender, distance and visibility show we are characterising user behaviour correctly in some respects. The lack of the expected effect of latency on gait makes it difficult to draw any conclusions however. It may be the effect of latency is hidden in the noise, but also likely is that our latency conditions are too low to elicit an effect. Samaraweera et al. had latencies of 75-225 ms, whereas our highest condition was ~28 ms. Future investigators should note the importance of high quality tracking and predictable behaviour patterns when considering gait.

Anecdotally, participants were asked during debrief if they noticed anything about the room change (e.g. shape or size). None mentioned latency or anything that would indicate they perceived the latency (e.g. the swimming effect). Experienced members of our lab could distinguish between all the latency conditions with ease, although they experienced the different conditions immediately next to each other, whereas the participants completed a blind walking trial during the change in latency. Fewer participants than expected remarked on the low graphical fidelity. One mentioned that the replica room was 'like a film set'. One participant asked directly if the synthetic room was modelled on a real place.

Obviously we cannot completely separate cognitive vs. functional systems, any more than we can consider only spatial or dynamic cues. All work together to form the perception of distance. Fink et al. suggest that there could be multiple causes [55]. These results do suggest however that distance compression may lie in the design of the VE - i.e. place illusion and plausibility illusion [207], rather than basic mechanical interference, regardless of *how* the physiomotor system is affected.

The differences between our VE, those used in previous studies, and commercial systems are subtle, but non-trivial. For example, our system does not use prediction. We would not have been able to run the experiment to dismiss latency without our custom system, but if our hypothesis is correct (that the significant factor is the design of the environment, rather than the implementation of the system) these results should be repeatable with an off-the-shelf commercial system such as the HTC Vive.

7.7 Summary

In the study described we examined the effects of latency on distance estimation compression in VR. Distance compression is a well-known phenomena that has been thoroughly studied. Despite this no single factor or model has been found to explain it. While some factors have been shown to be significant, they are not comprehensive. We suggested that it is worth exploring dynamic cues, in addition to static cues, as they are unlikely to function in isolation. Specifically, we suggest latency as a potential explanation. Latency is known to affect visuomotor functionality, and as an consequence of the inherent processing and transport delays in current computer systems, will have been present in all previous studies.

We constructed a VE system around a bespoke image generator. Our image generator was a purpose built real-time ray caster which computed pixels just-in-time. Coupled with a state-of-the-art tracking system our VE achieved a base latency of $1-4ms$. This the true latency, without techniques such as predictive warping. Interrante et al. discovered that distance compression disappears when a participant is virtually co-located in a replica of their real-world location. Based on this we designed a protocol that would provide us with samples of distance estimation accuracy and gait, under conditions with and without distance compression.

Our results were surprising and significant, if not as conclusive as we had hoped. Using a series of statistical tests of increasing power, we examined participant behaviour and found no effect of latency on distance judgement or physiomotor behaviour between 3-28 ms. However, we also failed to find any difference between the real and replica VEs. This is significant as our results indicate distance judgements are equally accurate in each. It is impossible to prove a negative, so we run our measures through a number of tests to ensure that they are accurate characterisations of user behaviour, and are not simply hiding the expected effect in the noise. We confirm our results overlap with previous studies, that where there is an effect it is plausible, and that the probability of making a Type II error is < 0.05. We are left to conclude that the deviation is a consequence of truly novel conditions, and not a protocol failure.

It is unfortunate our gait measures did not prove more conclusive. It is possible that our measures simply did not vary much between conditions. The borderline significance of some predictors, and the small effect size of others however, suggest that the measures contained more noise than planned. This may be alleviated in future works by adjusting the protocol to ensure more consistent behaviour, amenable to automated processing.

The single conclusion however is significant for two reasons. It demonstrates that it may be possible to build an entirely synthetic VE that facilitates equivalent distance estimation accuracy to those performed in the best conditions found so far. It also offers potential evidence that the underlying explanation for compression itself is a function of plausibility and believability - of the virtual environment, rather than the virtual reality system. We could not have run this study without our very high performance VE system. From our conclusions however we expect that these results should be repeatable with an unmodified commercial system such as the HTC Vive.

If they are, the implications will be of significant consequence for designers, suggesting it is possible to facilitate accurate distance judgement by manipulating only the content of a virtual scene.

Chapter 8

Conclusions

8.1 Overall Summary

The goal of this project was to explore the application of dataflow architectures to rendering for virtual reality. Visual stimuli are highly significant for actively interacting with the real-world, or even just existing within it, e.g. for balance. It is equally so for VR. This project has been concerned specifically with VR - virtual worlds that occlude the real world entirely. Many of the requirements of rendering for VR however apply to any synthetic environment. The objective is to render virtual stimuli that enables formation of the sensorimotor loop; to have users respond to the virtual stimuli, as they would real stimuli. Characteristics that affect this include spatial and visual qualities such as colour gamut and resolution, but also temporal behaviour, caused by latency and jitter.

Latency is an important characteristic of any computer interface that aims to facilitate direct interaction, but it is especially so for synthetic environments. Latency frustrates interaction, reducing efficiency and undermining the applications for both traditional interfaces and synthetic environments. In synthetic environments however it results in additional second-order effects, such as physical discomfort and negative training.

Latency was the focus of this project, because the characteristics required for minimising latency correspond closely with the abilities of dataflow computers. Dataflow computing has high throughput, and high determinism. It is highly inflexible however, and is limited in the computational complexity supported. That is, it is an ideal architecture for the implementation of a highly constrained piece of functionality that is the bottleneck in a larger system. Examples of such functions would be the triangle rasterisers and texture samplers on traditional GPUs. The potential to hardware accelerate a small, constrained problem has many applications in computer graphics. Tree traversal or ray-intersection tests are obvious examples. These are suitable for dataflow implementation, and indeed they should be investigated in the future. For minimising latency however, the biggest bottlenecks are in the end stages of the rendering pipeline - specifically the frame-based nature of the painters algorithm and scan-out, so it is this area that received the most focus.

We explored the feasibility of applying dataflow computing to rendering by building novel renderers. It was apparent to begin with that any renderer would have to directly drive the display. With the very simple scenes GPUs can already achieve a latency down to around the scan-out time of the display. Simply deriving an architecture capable of driving a display reliably was a challenge. Dataflow graphs are deterministic and high throughput in theory, but in practice implementation details leave a potentially significant discrepancy between their achieved characteristics, and the requirements of a typical ‘dumb’ line-scan display. Additional challenges arose as the platforms on which the algorithms would be implemented did not support driving displays, and had to be reverse engineered and modified to build this functionality into them.

We produced two rendering prototypes, a 2D sprite renderer and a 3D texture mapped primitive renderer. It was apparent early on that the best use of the dataflow architecture was to produce an image-based renderer - one that relied on integration of samples, rather than the transportation of light. This is because there is far more data and operation coherency in implementations the former, and dataflow implementations run at peak efficiency when the algorithm and data are coherent and localised. Authors have proposed a number of image-based architectures, with the one of the most extreme examples being a light field renderer. As the capabilities of the dataflow platform with respect to these algorithms was unpredictable, development of the prototypes were conservative, increasing complexity of the geometric proxies involved until we ended up with an advanced version of Regan & Pose's Address Recalculation Pipeline [180]. Our renderers had very low latencies, matching or exceeding any state-of-the-art. We proved motion-to-photon latencies of no more than 1 ms, but the actual rendering time was likely far lower. Our systems were comparable with those of Lincoln et al. [119] and Jota et al. [98] in terms of performance, but our implementation was far more flexible, supporting in our final experiment a fully immersive virtual world with 1:1 locomotion.

Numerous studies have investigated latency and physical interaction. A smaller number have investigated latency and presence or perception. That latency degrades user experience in computer mediated systems is not controversial. What is unproven is the levels at which it begins to have an effect. Understanding these thresholds may have as much practical significance as understanding the effect. As performance is pushed further and further the costs of each small gain increase. Identifying a basic set of requirements and tolerances can help focus resources to where they are required. Most authors hypothesised that the effects of latency would be linear down to zero, based on the observation that the effects of latency do not follow Weber's Law. We conducted two studies examining latencies at levels lower than any previous works, and in both cases our results were unexpected. For low level physical interaction we found that the effect of latency was non-linear. We hypothesised that this was due to an unexpected interaction between the underlying motor processes and the task itself. This result has significant implications for those using low level motor tasks to test the effects of latency. This is because it implies using only the typical metric of movement time may hide more nuanced, but significant, effects of latency. We also tested a perceptual functionality - distance estimation. At first glance latency would be a surprising predictor to analyse in the context of distance estimation. Synthetic environments significantly affect distance estimation however, and the cause has proved elusive over many studies. We were unable to find a significant effect of latency, but were able to reproduce distance judgements as accurate as any before seen in a synthetic environment. This is significant, because our conditions deviated far from those in which this was previously demonstrated.

8.2 Future of dataflow computing in rendering

Our prototypes show that dataflow architectures have the potential to play a significant role in the reduction of perceived latency. Their true-parallel nature makes them good replacements for a single shader running on a SIMD core in theory, but in practice the inflexibility of current spatial platforms (FPGAs) makes this use-case infeasible in real systems. A more immediately beneficial application of the technology would be to perform advanced versions of the functionality in devices such as the Warper Board [28] or Address Recalculation Pipeline [180].

The current algorithm used for real-time rendering on most GPUs - the painter's algorithm - is highly computationally efficient because few computations are expended on fragments that do not contribute to the final frame. However, when considering this algorithm it the context of the lifetime of the application it is very inefficient. Fragment computations are discarded each frame - 90 times a second or more, regardless of their temporal coherence. While GPUs are increasing in computational power, the future requirements of VR HMDs will be very strenuous [26]. It is possible in the future that the painters algorithm could be augmented with additional stages, which take advantage of this spatial coherence. It is also possible such techniques will be combined with a cascade of image warping stages, as suggested by Zheng et al. [263] and Lincoln et al. [119]. In the future a flexible SIMD-core based GPU could render to a data structure more persistent than frames but less persistent than geometry. This scene representation could be rasterised with a faster, more highly constrained local loop, maximising the useful lifetime of each computation, and reducing the apparent latency. Examples of such pipelines already exist, in the form of the Address Recalculation Pipeline [180], or the the virtual light field renderers [150]. Other uses of dataflow implementations applied to computer graphics could be found in the pipeline proposed by Jota et al. [98] which accelerates the apparent response of the GUI on mobile devices, and of course in offline applications such as ray tracing.

8.3 Rendering on Maxeler DFEs

The Maxeler Coria and Isca DFEs were a good platform to test the implementation of rendering dataflow algorithms, due to the very high grade FPGAs powering the cards. Designed for high performance computing installations however such cards would be unsuitable for use in final installations due to the cost of the FPGAs. Once a prototype algorithm is refined, the resources required can be identified with a high degree of confidence, and a more appropriate platform sourced. The other impediment to the use of DFEs are the interfaces. In order to directly drive a display, the infrastructure logic on the FPGA, and the physical interface of the cards themselves, needed to be modified. This took a considerable amount of time and was an entirely sunk cost, as these cards are unlikely to be used for rendering outside the apparatus presented here.

The use of a spatial programming language such as MaxJ is the biggest benefit of using a pre-existing platform such as a DFE. Rendering algorithms such as those described contain many complex functionalities, such as floating point operations, vector math, clock-domain crossing and backpressure signalling. The implementation of such design primitives on FPGAs is well studied and so is low-risk, but significantly increases the implementation time for a new design. The MaxCompiler toolchain handles this automatically. That considered, the lack of low level visibility can also be a problem during prototyping. When driving COTS display, if the output is out of specification, the display will simply fail to operate with little to no feedback on the source of the problem. MaxCompiler has a set of debugging tools which can capture the state of the graph, but most performance issues are transient and beyond their scope.

Dataflow graphs are deterministic in theory, but in practice implementation details can make them perform sub-optimally. One of the most egregious examples is the stalling behaviour. Resources such as memory are non-deterministic for example, even if the variance in response time is low. For this reason buffering must be placed around the resources, and the graph run at a higher rate than the eventual sink (the display) in order to smooth out irregularities. As the graph runs at a higher rate than data is consumed, it must be periodically throttled (stopped). The signal to stop all the nodes in the graph however takes a specific number of cycles to propagate back to deactivate all the nodes, and the same number to re-active them. This spin-up and spin-down time results in unavoidable gaps in the data stream and if the stream is started and stopped too often, the total throughput of the graph will drop below that required by the display. This is unavoidable in almost any practical dataflow implementation.

8.4 Latency experiments after commercial HMDs reach 20 ms

In previous decades, many experiments were limited to latencies of 20-40 ms or more, and so behaviour at latencies below this could only be hypothesised or extrapolated. Recent commercial VR systems are advertising latencies of between 10-20 ms however, and some are claiming imperceptible latencies using techniques such as predictive warping.

In our study on distance estimation, the conditions were chosen not to go above 20 ms partly due to participant comfort, but also because if an effect were demonstrated at these levels, it would in some senses be meaningless, since VR systems are no longer built which have such high latencies. It could be argued that in this case it would have been worthwhile, as it would demonstrate the experiment worked. As user studies are expensive to conduct however it is worth considering what advantages there are of continuing to test at such high levels. In the specific case of distance estimation, it probably is, because only one other study showed an effect of latency, and that was on gait, rather than distance judgements. In the case of physical interaction however, there is no question of the effect of latency, only at what levels, and how.

A more pertinent avenue of investigation may be into the effects of distortion and re-projection, rather than latency itself. Commerical HMD manufacturers can achieve 20 ms of true latency easily now that the screens powering such devices run at 90 Hz or more. Those that claim 10 ms or less are doing so with predictive warping techniques, and getting more and more ambitious in attempts to correct for longer and longer periods [199]. Such techniques will introduce errors, and the significance of these errors will depend on scene content, the nature of the motion, and perhaps even display technology/rendering technique as described in Section 5.3.2. The most impactful future studies may not be looking at behavioural thresholds of latency levels, but of visual artefact severity.

8.5 Future Works

Compensating for latency in VR is difficult. Latency is an inherent characteristic of the assembly of components that make up VR systems. Given current technology it cannot be eliminated, only minimised and compensated for. The computer systems that make up VEs are already state-of-the-art, due to the number and resolution of modalities requiring virtual stimuli. Each small gain in performance then becomes more and more expensive. At the same time, the nature of GPUs will change to support more GPGPU applications, to spread the cost over a larger number of industries, and the use of mobile VR will increase, in which computational power is already severely limited and not optimised for latency.

Given that computer graphics is a continuous trade-off between fidelity, computational power and memory, it is important to understand which aspect of the visual stimuli has largest effect on user experience. Some of the most effective immediate performance gains could be had by refactoring the existing rasteriasation pipeline to take advantage of temporal coherence. The current pipeline, by the time it reaches the GPU, does not use this anywhere. Reducing the computational power required to draw a frame will have benefits for VR systems at all levels. This could be done by making more invasive changes to the pipeline, inserting another stage between geometry and fragments. Alternatively it could be augmented with additional stages on the end, with advanced image warping facilitating reduced frame rates.

The latter solution is attractive due to its simplicity, but the implications of making ever increasingly extreme distortions to images in terms of user experience are unknown. Synthesising to a more complete data structure that can be rendered from in a fast local loop will help reduce the potential for distortions. It will require much more development however than simple image warping architectures. By manipulating the tracking data fed to the DFE, our realtime raycaster can emulate various rendering approaches & image warping techniques, as well as various latencies. The question to examine in future works is, which is it most productive to optimise?

Appendix A

Publications

This project has resulted in the following publications, appearing in or submitted to peer-reviewed conferences and journals.

FRISTON, S., STEED, A., TILBURY, S., AND GAYDADJIEV, G. Ultra low latency dataflow renderer. In *25th International Conference on Field Programmable Logic and Applications, FPL 2015* (2015)
Contains extracts of work presented in Chapter 3.

FRISTON, S., STEED, A., TILBURY, S., AND GAYDADJIEV, G. Construction and Evaluation of an Ultra Low Latency Frameless Renderer for VR. *IEEE Transactions on Visualization and Computer Graphics 22*, 4 (apr 2016), 1377–1386
Contains extracts of work presented in Chapters 4 & 5.

FRISTON, S., KARLSTROM, P., AND STEED, A. The Effects of Low Latency on Pointing and Steering Tasks. *IEEE Transactions on Visualization and Computer Graphics 22*, 5 (may 2016), 1605–1615
Contains extracts of work presented in Chapter 6.

FRISTON, S., AND STEED, A. Measuring Latency in Virtual Environments. *IEEE Transactions on Visualization and Computer Graphics (Proceedings Virtual Reality 2014) 20*, 4 (2014)

Sebastian Friston, Simon Tilbury, Georgi Gaydadjiev, and Anthony Steed. Latency, Gait and Distance Compression - or Lack of - in Virtual Reality.
Under Review.
Contains extracts of work presented in Chapter 7.

Appendix B

List of Acronyms

AR	Augmented Reality
CAVE	Cave Automatic Virtual Environment
CG	Computer Generated
COTS	Commerical Off-The-Shelf
DFE	Dataflow Engine
DMD	Digital Micro-mirror Device
FOV	field-of-view
FPGA	Field Programmable Gate Array
HMD	head mounted display
HVS	Human Visual System
ID	Index of Difficulty
IQM	Image Quality Measure
IMU	Inertial Measurement Unit
IQA	Image Quality Assessment
IQM	Image Quality Measure

MR	Mixed Reality
MSE	Mean-Squared Error
MT	Movement Time
OLED	Organic Light-Emitting Diode
PARTS	Programmable And Reconfigurable Tool Set
PCML	Pseudo Current Mode Logic
PCS	Physical Coding Sublayer
PMA	Physical Media Access
PSF	Parallel Sub-Field
PSNR	Peak Signal to Noise Ratio
RMS	Root Mean Squared
SERDES	Serialisation-Deserialisation
SIMD	Single Instruction Multiple Data
TMDS	Transition Minimized Differential Signalling
VE	Virtual Environment
VIF	Visual Information Fidelity
VITE	Vector Integration to Endpoint
VLF	Virtual Light Field
VR	Virtual Reality

Appendix C

Colophon

This document was set using LaTeX and BibTeXwith the UCL Thesis document class, composed with TexMaker and the following tools.

Mendeley. Matlab. Microsoft Office Visio. Gimp. Notepad++.

Bibliography

[1] ACCOT, J., AND ZHAI, S. Beyond Fitts' Law: Models for Trajectory-Based HCI Tasks. In *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '97* (New York, New York, USA, 1997), ACM Press, pp. 295–302.

[2] ADAM, J. J., MOL, R., PRATT, J., AND FISCHER, M. H. Moving farther but faster: an exception to Fitts's law. *Psychological science 17*, 9 (sep 2006), 794–8.

[3] ADELSON, E. H., AND BERGEN, J. R. The Plenoptic Function and the Elements of Early Vision. *Computational Models of Visual Processing* (1991), 3–20.

[4] ADELSTEIN, B. D., BURNS, E. M., ELLIS, S. R., AND HILL, M. I. Latency Discrimination Mechanisms in Virtual Environments: Velocity and Displacement Error Factors. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* (sep 2005), vol. 49, pp. 2221–2225.

[5] ADELSTEIN, B. D., LEE, T. G., AND ELLIS, S. R. Head Tracking Latency in Virtual Environments: Psychophysics and a Model. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* (oct 2003), vol. 47, pp. 2083–2087.

[6] ALLISON, R. S., HARRIS, L. R., JENKIN, M., JASIOBEDZKA, U., AND ZACHER, J. E. Tolerance of temporal delay in virtual environments. In *Pro-*

ceedings of the 2001 IEEE Virtual Reality Conference (2001), IEEE Computer Society, pp. 247–254.

[7] ALTERA. The Evolution of High-Speed Transceiver Technology. Tech. Rep. November, Altera, 2002.

[8] ALTERA. *Stratix V Device Overview*. 2014.

[9] ANVARI, M., BRODERICK, T., STEIN, H., CHAPMAN, T., GHODOUSSI, M., BIRCH, D. W., MCKINLEY, C., TRUDEAU, P., DUTTA, S., AND GOLDSMITH, C. H. The impact of latency on surgical precision and task completion during robotic-assisted remote telepresence surgery. *Computer Aided Surgery: Official Journal of the International Society for Computer Aided Surgery 10*, 2 (jan 2005), 93–99.

[10] ARCHDEACON, J. L., IWAI, N., AND SWEET, B. T. Designing and Developing an Image Generator for the Operational Based Vision Assessment Simulator. *American Institute of Aeronautics and Astronautics 1* (2012), 436–450.

[11] ATHAVALE, A., AND CHRISTENSEN, C. *High-Speed Serial I/O Made Simple*, 1.0 ed. Xilinx, 2005.

[12] BAKER, R. Temporal spatial data, the gait cycle and gait graphs Temporal spatial parameters. In *An Introduction to Clinical Gait Analysis*. University of Salford.

[13] BANTON, T., STEFANUCCI, J., DURGIN, F., FASS, A., AND PROFFITT, D. The Perception of Walking Speed. *Presence 14*, 4 (2005), 394–406.

[14] BEAMISH, D., BHATTI, S., CHUBBS, C. S., MACKENZIE, I. S., WU, J., AND JING, Z. Estimation of psychomotor delay from the Fitts' law coefficients. *Biological Cybernetics 101*, 4 (oct 2009), 279–96.

[15] BEAMISH, D., BHATTI, S., WU, J., AND JING, Z. Performance limitations from delay in human and mechanical motor control. *Biological Cybernetics 99*, 1 (jul 2008), 43–61.

[16] BEELER, D., AND GOSALIA, A. Asynchronous timewarp on oculus rift. `https://developer3.oculus.com/blog/asynchronous-timewarp-on-oculus-rift/`, 2017.

[17] BENTE, G., RÜGGENBERG, S., KRÄMER, N. C., AND ESCHENBURG, F. Avatar-mediated networking: Increasing social presence and interpersonal trust in net-based collaborations. *Human Communication Research 34*, 2 (2008), 287–318.

[18] BERGAMASCO, M., ALLOTTA, B., BOSIO, L., FERRETTI, L., PARRINI, G., PRISCO, G., SALSEDO, F., AND SARTINI, G. An arm exoskeleton system for teleoperation and virtual environments applications. In *Proceedings of the 1994 IEEE International Conference on Robotics and Automation* (1994), pp. 1449–1454.

[19] BERGMAN, L., FUCHS, H., GRANT, E., AND SPACH, S. Image rendering by adaptive refinement. In *Proceedings of ACM SIGGRAPH '86* (Dallas, 1986), vol. 20, pp. 29–37.

[20] BIRKLBAUER, C., OPELT, S., AND BIMBER, O. Rendering Gigaray Light Fields. *Eurographics 32*, 2 (may 2013), 469–478.

[21] BLINCH, J., CAMERON, B. D., HODGES, N. J., AND CHUA, R. Do preparation or control processes result in the modulation to Fitts' law for movements to targets with placeholders? *Experimental brain research 223*, 4 (dec 2012), 505–15.

[22] BOOTSMA, R., FERNANDEZ, L., AND MOTTET, D. Behind Fitts' law: kinematic patterns in goal-directed movements. *International Journal of Human-Computer Studies 61*, 6 (dec 2004), 811–821.

[23] BOTZER, L., AND KARNIEL, A. Feedback and feedforward adaptation to visuomotor delay during reaching and slicing movements. *The European Journal of Neuroscience 38*, 1 (jul 2013), 2108–23.

[24] BOULOS, S., WALD, I., AND SHIRLEY, P. Geometric and Arithmetic Culling Methods for Entire Ray Packets. Tech. rep., University of Utah, 2006.

[25] BOWLES, H., MITCHELL, K., SUMNER, R. W., MOORE, J., AND GROSS, M. Iterative Image Warping. *Computer Graphics Forum 31*, 2pt1 (may 2012), 237–246.

[26] BRENNAN, D. Oculus chief scientist predicts the next 5 years of vr technology. `http://www.roadtovr.com/michael-abrash-explores-next-5-years-vr-technology/`, 2016.

[27] BUEHLER, C., BOSSE, M., MCMILLAN, L., GORTLER, S., AND COHEN, M. Unstructured lumigraph rendering. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques - SIGGRAPH '01* (New York, New York, USA, 2001), ACM Press, pp. 425–432.

[28] BUKER, T. J., VINCENZI, D. A., AND DEATON, J. E. The Effect of Apparent Latency on Simulator Sickness While Using a See-Through Helmet-Mounted Display: Reducing Apparent Latency With Predictive Compensation. *Human Factors: The Journal of the Human Factors and Ergonomics Society 54*, 2 (jan 2012), 235–249.

[29] BURGERS, W. *Tile-Based Rendering*. PhD thesis, Technische Universiteit Eindhoven, 2005.

[30] BUXTON, W. A. S. Telepresence: Integrating Shared Task and Person Spaces. In *Proceedings of the Conference on Graphics Interface (GI'92)* (1992), pp. 123–129.

[31] ČADÍK, M., HERZOG, R., MANTIUK, R., MYSZKOWSKI, K., AND SEIDEL, H.-P. New Measurements Reveal Weaknesses of Image Quality Metrics in

Evaluating Graphics Artifacts. *ACM Transactions on Graphics 31*, 6 (2012), Article 147.

[32] CAMPOS, J., NUSSECK, H.-G., WALLRAVEN, C., MOHLER, B., AND BÜLTHOFF, H. Visualization and (mis)perceptions in virtual reality. In *Proceedings of Workshop Sichtsysteme* (Aachen, Germany, 2007), pp. 10–14.

[33] CAMPOS, J. L., BUTLER, J. S., AND BULTHOFF, H. H. Multisensory integration in the estimation of walked distances. *Experimental Brain Research 218*, 4 (2012), 551–565.

[34] CATMULL, E. E. *A Subdivision Algorithm For Computer Display of Curved Surfaces*. PhD thesis, University of Utah, 1974.

[35] CHALMERS, A., DEBATTISTA, K., MASTOROPOULOU, G., AND PAULO DOS SANTOS, L. There-reality: selective rendering in high fidelity virtual environments. *The International Journal of Virtual Reality 6*, 1 (2007), 1–10.

[36] CHUNG, K. M., AND SO, R. H. Effects of Hand Movement Lag on Discrete Manual Control Tasks in Virtual Environements. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* (sep 1999), vol. 43, pp. 1210–1213.

[37] CLARK, J. H. A fast scan-line algorithm for rendering parametric surfaces. In *Proceedings of the 6th annual conference on Computer graphics and interactive techniques - SIGGRAPH '79* (New York, New York, USA, 1979), ACM Press, p. 174.

[38] CRUZ-NEIRA, C., SANDIN, D. J., DEFANTI, T. A., KENYON, R. V., AND HART, J. C. The CAVE: audio visual experience automatic virtual environment. *Communications of the ACM 35*, 6 (1992), 64–72.

[39] CUNNINGHAM, A. Shedding some realistic light on imagination's real-time ray tracing card. `http://arstechnica.com/gadgets/2013/01/`

shedding-some-realistic-light-on-imaginations-rea l-time-ray-tracing-card/, 2013.

[40] CUTLER, AND DURAND. MIT 6.837 Lecture Notes - Monte-Carlo Ray Tracing, 2014.

[41] DAYAL, A., WOOLLEY, C., WATSON, B., AND LUEBKE, D. Adaptive frameless rendering. *ACM SIGGRAPH 2005 Courses* (2005).

[42] DEBATTISTA, K., CHALMERS, A., GILLIBRAND, R., LONGHURST, P., MASTOROPOULOU, G., AND SUNDSTEDT, V. Parallel selective rendering of high-fidelity virtual environments. *Parallel Computing 33*, 6 (2007), 361–376.

[43] DESHMUKH, S. P. DESENSITISATION OF AIRSICKNESS IN TRAINEE PILOTS BY PHYSICAL EXERCISE THERAPY. *Indian Journal of Aerospace Medicine Special Co* (2007), 37–42.

[44] DI LUCA, M. New Method to Measure End-to-End Delay of Virtual Reality. *Presence 19*, 6 (dec 2010), 569–584.

[45] DIDYK, P., EISEMANN, E., RITSCHEL, T., MYSZKOWSKI, K., AND SEIDEL, H.-P. Perceptually-motivated Real-time Temporal Upsampling of 3D Content for High-refresh-rate Displays. *Computer Graphics Forum 29*, 2 (may 2010), 713–722.

[46] DIDYK, P., RITSCHEL, T., EISEMANN, E., MYSZKOWSKI, K., AND SEIDEL, H.-P. Adaptive image-space stereo view synthesis. In *Proceedings of the Vision, Modeling, and Visualization Workshop (2010)* (2010), pp. 299–306.

[47] DIGITAL DISPLAY WORKING GROUP. Digital Visual Interface Specification.

[48] DOVE, K. Virtual reality reaches 100-million pixels with nvidia quadro® technology. http://www.nvidia.com/object/io_1193136933888.html, 2007.

[49] DURGIN, F. H., REED, C., AND TIGUE, C. Step frequency and perceived self-motion. *ACM Transactions on Applied Perception 4*, 1 (2007).

[50] DURLACH, N. I., AND MAVOR, A. S. *Virtual Reality: Scientific and Technological Challenges*. 1994.

[51] ELLIOTT, D., HELSEN, W. F., AND CHUA, R. A century later: Woodworth's (1899) two-component model of goal-directed aiming. *Psychological Bulletin 127*, 3 (2001), 342–357.

[52] ELLIS, S. R., MANIA, K., ADELSTEIN, B. D., AND HILL, M. I. Generalizeability of Latency Detection in a Variety of Virtual Environments. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* (sep 2004), vol. 48, pp. 2632–2636.

[53] ELLIS, S. R., YOUNG, M. J., ADELSTEIN, B. D., AND EHRLICH, S. M. Discrimination of changes of latency during voluntary hand movement of virtual objects. In *Proceedings of the Human Factors and Ergonomics Society* (1999).

[54] FERWERDA, J. A. Three varieties of realism in computer graphics. In *Proceedings of SPIE Human Vision and Electronic Imaging '03* (2003), pp. 290–297.

[55] FINK, P. W., FOO, P. S., AND WARREN, W. H. Obstacle avoidance during walking in real and virtual environments. *ACM Transactions on Applied Perception 4*, 1 (2007).

[56] FITTS, P. M. The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology 47*, 6 (1954), 381–391.

[57] FORTENBAUGH, F. C., CHAUDHURY, S., HICKS, J. C., HAO, L., AND TURANO, K. A. Gender differences in cue preference during path integration in virtual environments. *ACM Transactions on Applied Perception 4*, 1 (2007).

[58] FRANZ, G., VON DER HEYDE, M., AND BÜLTHOFF, H. H. An empirical approach to the experience of architectural space in virtual reality-exploring relations between features and affective appraisals of rectangular indoor spaces. *Automation in Construction 14*, 2 SPEC. ISS. (2005), 165–172.

[59] FRENZ, H., LAPPE, M., KOLESNIK, M., AND BÜHRMANN, T. Estimation of travel distance from visual motion in virtual environments. *ACM Transactions on Applied Perception 4*, 1 (2007).

[60] FRIMENKO, R., WHITEHEAD, C., AND BRUENING, D. Do Men and Women Walk Differently? A Review and Meta-Analysis of Sex Difference in Non-Pathological Gait Kinematic. Tech. rep., INFOSCITEX CORP DAYTON OH, 2014.

[61] FRISTON, S., KARLSTROM, P., AND STEED, A. The Effects of Low Latency on Pointing and Steering Tasks. *IEEE Transactions on Visualization and Computer Graphics 22*, 5 (may 2016), 1605–1615.

[62] FRISTON, S., AND STEED, A. Measuring Latency in Virtual Environments. *IEEE Transactions on Visualization and Computer Graphics (Proceedings Virtual Reality 2014) 20*, 4 (2014).

[63] FRISTON, S., STEED, A., TILBURY, S., AND GAYDADJIEV, G. Ultra low latency dataflow renderer. In *25th International Conference on Field Programmable Logic and Applications, FPL 2015* (2015).

[64] FRISTON, S., STEED, A., TILBURY, S., AND GAYDADJIEV, G. Construction and Evaluation of an Ultra Low Latency Frameless Renderer for VR. *IEEE Transactions on Visualization and Computer Graphics 22*, 4 (apr 2016), 1377–1386.

[65] GANACIM, F., FIGUEIREDO, L. H., AND NEHAB, D. Beam Casting Implicit Surfaces on the GPU with Interval Arithmetic. In *Proceedings of the 24th SIBGRAPI Conference on Graphics, Patterns and Images* (aug 2011), pp. 72–77.

[66] GARAU, M., SLATER, M., PERTAUB, D.-P., AND RAZZAQUE, S. The Responses of People to Virtual Humans in an Immersive. *Presence 14*, 1 (2005), 104–116.

[67] GORTLER, S. J., GRZESZCZUK, R., SZELISKI, R., AND COHEN, M. F. The lumigraph. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques - SIGGRAPH '96* (New York, New York, USA, 1996), ACM Press, pp. 43–54.

[68] GROSSMAN, G. E., LEIGH, R. J., ABEL, L. A., LANSKA, D. J., AND THURSTON, S. E. Frequency and velocity of rotational head perturbations during locomotion. *Experimental Brain Research 70*, 3 (1988), 470–476.

[69] GUTWIN, C. The Effects of Network Delays on Group Work in Real-Time Groupware. In *Proceedings of the Seventh European Conference on Computer-Supported Cooperative Work* (Bonn, Germany, 2001), pp. 299–318.

[70] HALLER, I., AND BARUCH, Z. F. High-speed clock recovery for low-cost FPGAs. In *Proceedings of the 2010 Design, Automation & Test in Europe Conference & Exhibition (DATE 2010)* (mar 2010), IEEE, pp. 610–613.

[71] HARRIS, M. Maxwell: The most advanced cuda gpu ever made. `http://devblogs.nvidia.com/parallelforall/maxwell-most-advanced-cuda-gpu-ever-made`, 2014.

[72] HARVEY, M. A., AND SANCHEZ-VIVES, M. V. The Binding Problem in Presence Research. *Presence: Teleoperators and Virtual Environments 14*, 5 (oct 2005), 616–621.

[73] HE, D., LIU, F., PAPE, D., DAWE, G., AND SANDIN, D. Video-Based Measurement of System Latency. *International Immersive Projection Technology Workshop* (2000).

[74] HECKBERT, P. S., AND HANRAHAN, P. Beam tracing polygonal objects. *ACM SIGGRAPH Computer Graphics 18*, 3 (jul 1984), 119–127.

[75] HOFFMAN, H. G., CHAMBERS, G. T., MEYER, W. J., ARCENEAUX, L. L., RUSSELL, W. J., SEIBEL, E. J., RICHARDS, T. L., SHARAR, S. R., AND PATTERSON, D. R. Virtual reality as an adjunctive non-pharmacologic analgesic for acute burn pain during medical procedures. *Annals of Behavioral Medicine 41*, 2 (2011), 183–191.

[76] HOLLMAN, J. H., BREY, R. H., BANG, T. J., AND KAUFMAN, K. R. Does walking in a virtual environment induce unstable gait? An examination of vertical ground reaction forces. *Gait & posture 26*, 2 (jul 2007), 289–94.

[77] HOLLMAN, J. H., BREY, R. H., ROBB, R. A., BANG, T. J., AND KAUFMAN, K. R. Spatiotemporal gait deviations in a virtual reality environment. *Gait and Posture 23*, 4 (2006), 441–444.

[78] HRUSKA, J. The future of ray tracing, reviewed: Caustic's r2500 accelerator finally moves us towards real-time ray tracing. `http://www.extremetech.com/extreme/161074-the-future-of-ray-tracing-reviewed-caustics-r2500-accelerator-finally-moves-us-towards-real-time-ray-tracing`, 2013.

[79] HUDSON, L. Virtual reality's possibilities lure video game developers. `http://www.nytimes.com/2016/09/29/technology/personaltech/virtual-realitys-possibilities-lure-video-game-developers.html`, 2016.

[80] HUGHES, J. F., VAN DAM, A., MCGUIRE, M., SKLAR, D. F., FOLEY, J. D., FEINER, S. K., AND AKELEY, K. *Computer Graphics: Principles and Practice (3rd Edition)*, 3rd ed. Addison-Wesley, 2013.

[81] HUYS, R., FERNANDEZ, L., BOOTSMA, R. J., AND JIRSA, V. K. Fitts' law is not continuous in reciprocal aiming. In *Proceedings of the Royal Society B: Biological Sciences* (apr 2010), vol. 277, pp. 1179–1184.

[82] IHM, I., PARK, S., AND LEE, R. K. Rendering of spherical light fields. In *Proceedings of the Fifth Pacific Conference on Computer Graphics and Applications* (1997), IEEE Comput. Soc, pp. 59–68.

[83] IJSSELSTEIJN, W., AND RIVA, G. Being There : The experience of presence in mediated environments. In *Being There: Concepts, effects and measurement of user presence in synthetic environments*. Ios Press, Amsterdam, 2003.

[84] INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. ISO9241-9: Ergonomic requirements for office work with visual display terminals (VDTs) — Part 9 Requirements for non-keyboard input devices, 2000.

[85] INTERRANTE, V., RIES, B., AND ANDERSON, L. Distance Perception in Immersive Virtual Environments, Revisited. In *IEEE Virtual Reality Conference (VR 2006)* (2006), IEEE, pp. 3–10.

[86] INTERRANTE, V., RIES, B., LINDQUIST, J., KAEDING, M., AND ANDERSON, L. Elucidating Factors that Can Facilitate Veridical Spatial Perception in Immersive Virtual Environments. *Presence: Teleoperators and Virtual Environments 17*, 2 (apr 2008), 176–198.

[87] ISAKSEN, A., MCMILLAN, L., AND GORTLER, S. J. Dynamically reparameterized light fields. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques - SIGGRAPH '00* (New York, New York, USA, 2000), ACM Press, pp. 297–306.

[88] ITO, H., OGAWA, M., AND SUNAGA, S. Evaluation of an organic light-emitting diode display for precise visual stimulation. *Journal of Vision 13*, 7 (jun 2013), 1–21.

[89] JANKOWSKI, J., AND HACHET, M. A Survey of Interaction Techniques for Interactive 3D Environments. *Eurographics 2013 - State of the Art Reports* (2013), 65–93.

[90] JAX, S. A., ROSENBAUM, D. A., AND VAUGHAN, J. Extending Fitts' Law to manual obstacle avoidance. *Experimental brain research 180*, 4 (jul 2007), 775–9.

[91] JAY, C., GLENCROSS, M., AND HUBBOLD, R. Modeling the effects of delayed haptic and visual feedback in a collaborative virtual environment. *ACM Transactions on Computer-Human Interaction 14*, 2 (aug 2007).

[92] JELFS, A., AND WHITELOCK, D. The notion of presence in virtual learning environments: What makes the environment "real". *British Journal of Educational Technology 31*, 2 (2000), 145–152.

[93] JENSEN, H. W. Global Illumination using Photon Maps. Tech. rep., The Technical University of Denmark, 1996.

[94] JERALD, J., PECK, T., STEINICKE, F., AND WHITTON, M. Sensitivity to scene motion for phases of head yaws. In *Proceedings of the 5th symposium on Applied Perception in Graphics and Vsualization* (New York, 2008), ACM Press, p. 155.

[95] JERALD, J., AND WHITTON, M. Relating Scene-Motion Thresholds to Latency Thresholds for Head-Mounted Displays. In *Proceedings of the 2009 IEEE Virtual Reality Conference* (mar 2009), IEEE, pp. 211–218.

[96] JERALD, J., WHITTON, M., AND BROOKS, F. P. Scene-Motion Thresholds During Head Yaw for Immersive Virtual Environments. *ACM Transactions on Applied Perception 9*, 1 (2012), 4:2–4:23.

[97] JONES, J. A., SWAN, J. E., SINGH, G., REDDY, S., MOSER, K., HUA, C., AND ELLIS, S. R. Improvements in visually directed walking in virtual environments cannot be explained by changes in gait alone. In *Proceedings of the ACM Symposium on Applied Perception - SAP '12* (New York, New York, USA, 2012), no. October, ACM Press.

[98] JOTA, R., FORLINES, C., LEIGH, D., SANDERS, S., AND WIGDOR, D. Towards Zero-Latency User Experiences. Tech. rep., Tactual Labs Co.

[99] JOTA, R., NG, A., DIETZ, P., AND WIGDOR, D. How fast is fast enough? In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems - CHI '13* (New York, New York, USA, 2013), ACM Press, pp. 2291–2300.

[100] JUNE, X. Video Connectivity Using TMDS I / O in Spartan-3A FPGAs FPGA Throughput and Video Screen Modes Notes :. 1–36.

[101] JUNIPER. Understanding cos flow control (ethernet pause and pfc). `http://www.juniper.net/techpubs/en_US/junos13.2/topics/concept/cos-qfx-series-congestion-notification-understanding.html`, 2014.

[102] KAJIYA, J. T. The rendering equation. *ACM SIGGRAPH Computer Graphics 20*, 4 (aug 1986), 143–150.

[103] KEETH, B., AND BAKER, R. J. *DRAM Circuit Design: A Tutorial.* John Wiley & Sons, 2001.

[104] KHANNA, P., SLATER, M., MORTENSEN, J., AND YU, I. A Virtual Light Field for Propagation and Walkthrough of Globally Illuminated Scenes. Tech. rep., University College London, 2005.

[105] KIM, H., DIGIACOMO, T., EGGES, A., LYARD, E., AND GARCHERY, S. Believable virtual environment: Sensory and perceptual believability. *Workshop on Believability in Virtual Environments* (2008).

[106] KIM, S.-S., NAM, S.-W., AND LEE, I.-H. Fast ray-triangle intersection computation using reconfigurable hardware. Tech. rep., Digital Content Research Division, Heidelberg, 2007.

[107] KNOLL, A., HIJAZI, Y., KENSLER, A., SCHOTT, M., HANSEN, C., AND HAGEN, H. Fast Ray Tracing of Arbitrary Implicit Surfaces with Interval and Affine Arithmetic. *Computer Graphics Forum* (2008).

[108] KONTARINIS, D. A., HOWE, R. D., AND HALL, P. Tactile Display of Vibratory Information in Teleoperation and Virtual Environments. *Applied Sciences 4*, 4 (1995), 387–402.

[109] KULIKOV, S., MACKENZIE, I. S., AND STUERZLINGER, W. Measuring the effective parameters of steering motions. *CHI '05 extended abstracts on Human factors in computing systems - CHI '05* (2005), 1569.

[110] KUNZ, B. R., WOUTERS, L., SMITH, D., THOMPSON, W. B., AND CREEM-REGEHR, S. H. Revisiting the effect of quality of graphics on distance judgments in virtual environments: A comparison of verbal reports and blind walking. *Attention, Perception, & Psychophysics 71*, 6 (aug 2009), 1284–1293.

[111] LAPERE, S. John carmack: "eventually ray tracing will win". `http://raytracey.blogspot.co.uk/2011/08/john-carmack-eventually-ray-tracing.html`, 2011.

[112] LEBEDEV, A. S. History of global illumination algorithms. `http://lebedev.as/index.php?p=1_10_NEW-Articles`, 2011.

[113] LEVOY, M. Light fields and computational imaging. *IEEE Computer* (2006).

[114] LEVOY, M., AND HANRAHAN, P. Light Field Rendering. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques - SIGGRAPH '96* (New York, USA, 1996), ACM Press, pp. 31–42.

[115] LI, J., ZHOU, K., WANG, Y., AND SHUM, H.-Y. A Novel Image-Based Rendering System With A Longitudinally Aligned Camera Array. *Eurographics 2000 - Short Presentations* (2000).

[116] LI, X., LIU, B., AND WU, E. Full Solid Angle Panoramic Viewing by Depth Image Warping on Field Programmable Gate Array. *International Journal of Virtual Reality 6*, 2 (2007), 69–77.

[117] LIN, Q. L. Q., AND KUO, C. K. C. Virtual tele-operation of underwater robots. In *Proceedings of International Conference on Robotics and Automation* (1997), vol. 2, pp. 1022–1027.

[118] LIN, Z., AND SHUM, H.-Y. On the number of samples needed in light field rendering with constant-depth assumption. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition - CVPR 2000* (2000), vol. 1, IEEE Comput. Soc, pp. 588–595.

[119] LINCOLN, P., BLATE, A., SINGH, M., WHITTED, T., STATE, A., LASTRA, A., AND FUCHS, H. From Motion to Photons in 80 Microseconds: Towards Minimal Latency for Virtual and Augmented Reality. *IEEE Transactions on Visualization and Computer Graphics 22*, 4 (2016), 1367–1376.

[120] LIU, B., WEI, L.-Y., XU, Y.-Q., AND WU, E. Multi-layer depth peeling via fragment sort. In *Proceedings of the 2009 11th IEEE International Conference on Computer-Aided Design and Computer Graphics* (aug 2009), IEEE, pp. 452–456.

[121] LIU, L., AND LIERE, R. V. The Effect of Varying Path Properties in Path Steering Tasks. In *Proceedings of the 16th Eurographics conference on Virtual Environments & Second Joint Virtual Reality (EGVE - JVRC'10)* (2010), pp. 9–16.

[122] LIU, L., AND LIERE, R. V. Modelling Object Pursuit for Desktop Virtual Reality. *IEEE Transactions on Visualization and Computer Graphics 18*, 7 (2012), 1017–1026.

[123] LIU, L., MARTENS, J.-B., AND VAN LIERE, R. Revisiting path steering for 3D manipulation tasks. In *Proceedings of the 2010 IEEE Symposium on 3D User Interfaces (3DUI)* (mar 2010), Ieee, pp. 39–46.

[124] LONDON, U. C. Immersive virtual environments laboratory. `http://vr.cs.ucl.ac.uk/`, 2014.

[125] LOOMIS, J. M., DA SILVA, J. A., PHILBECK, J. W., AND FUKUSIMA, S. S. Visual Perception of Location and Distance. *Current Directions in Psychological Science 5*, 3 (jun 1996), 72–77.

[126] LUXION. Keyshot 4 manual: Material types and their settings. `https://www.keyshot.com/keyshot4/manual/material_types/high_res/metal_roughness.html`, 2014.

[127] MACKENZIE, I. S. A note on the information-theoretic basis for Fitts' Law. *Journal of Motor Behavior 3*, 21 (1989), 323–330.

[128] MACKENZIE, I. S. Movement Time Prediction in Human-Computer Interfaces. In *Readings in human-computer interaction* (1995), pp. 483–493.

[129] MACKENZIE, I. S., AND WARE, C. Lag as a determinant of human performance in interactive systems. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (New York, 1993), ACM Press, pp. 488–493.

[130] MANIA, K., ADELSTEIN, B. D., ELLIS, S. R., AND HILL, M. I. Perceptual sensitivity to head tracking latency in virtual environments with varying degrees of scene complexity. In *Proceedings of the 1st Symposium on Applied Perception in Graphics and Visualization* (2004), pp. 39–48.

[131] MANIA, K., TROSCIANKO, T., HAWKES, R., AND CHALMERS, A. Fidelity Metrics for Virtual Environment Simulations Based on Spatial Memory Awarness States. *Presence 12* (2003), 296–310.

[132] MARK, W. R., BISHOP, G., AND MCMILLAN, L. Post-Rendering Image Warping for Latency Compensation UNC-CH Computer Science Technical Report\# 96-020. *Science* (1996).

[133] MARK, W. R., MCMILLAN, L., AND BISHOP, G. Post-Rendering 3D Warping. In *Proceedings of the 1997 Symposium on Interactive 3D Graphics* (1997), pp. 7–16.

[134] MASHOUR, G. The cognitive binding problem: from Kant to quantum neurodynamics. *NeuroQuantology 2*, 1 (2004), 29–38.

[135] MAXELER TECHNOLOGIES LTD. *Acceleration Tutorial - Loops and Pipelining*. Maxeler Technologies, 2014.

[136] MAXELER TECHNOLOGIES LTD. *Dataflow Programming for Networking*. Maxeler Technologies, 2014.

[137] MAXELER TECHNOLOGIES LTD. *MaxCompiler Manager Compiler Tutorial*. Maxeler Technologies, 2014.

[138] MAXELER TECHNOLOGIES LTD. *Multiscale Dataflow Programming*. Maxeler Technologies, 2014.

[139] MAXFIELD, C. *FPGAs - Instant Access*. Elsevier, dec 2008.

[140] MCGUIRE, M., AND LUEBKE, D. Hardware-Accelerated Global Illumination by Image Space Photon Mapping. Tech. rep., Williams College and NVIDIA Corporation, 2001.

[141] MCNAMARA, A. Visual Perception in Realistic Image Synthesis. *Computer Graphics Forum 20*, 4 (2001), 211–224.

[142] MEEHAN, M., RAZZAQUE, S., INSKO, B., WHITTON, M., AND BROOKS, F. P. Review of four studies on the use of physiological reaction as a measure of presence in stressful virtual environments. *Applied Psychophysiology Biofeedback 30*, 3 (2005), 239–258.

[143] MEEHAN, M., RAZZAQUE, S., WHITTON, M. C., AND BROOKS JR., F. P. Effect of latency on presence in stressful virtual environments. In *Proceedings of the 2003 IEEE Virtual Reality Conference* (2003).

[144] MESETH, J., MÜLLER, G., AND KLEIN, R. Surface Light Field Rendering for Virtual Product Design. In *Proceedings of The 11th International Conference on Virtual Systems and Multimedia (VSMM 2005)* (2005), pp. 185–188.

[145] MEYER, D. E., ABRAMS, R. A., KORNBLUM, S., WRIGHT, C. E., AND SMITH, J. E. Optimality in human motor performance: ideal control of rapid aimed movements. *Psychological review 95*, 3 (1988), 340–370.

[146] MILLER, J. D., ANDERSON, M. R., WENZEL, E. M., AND MCCLAIN, B. U. Latency measurement of a real-time virtual acoustic environment rendering system. In *Proceedings of the 2003 International Conference on Auditory Display* (Boston, MA, 2003).

[147] MINE, M. R. Characterization of end-to-end delays in head-mounted display systems. Tech. rep., University of North Carolina at Chapel Hill, 1993.

[148] MOHLER, B. J., CAMPOS, J. L., WEYEL, M. B., AND BÜLTHOFF, H. H. Gait parameters while walking in a head-mounted display virtual environment and the real world. In *Proceedings of the 13th Eurographics Symposium on Virtual Environments* (2007), pp. 85–88.

[149] MOHLER, B. J., CREEM-REGEHR, S. H., AND THOMPSON, W. B. The influence of feedback on egocentric distance judgments in real and virtual environments. In *Proceedings of the 3rd symposium on Applied perception in graphics and visualization - APGV '06* (New York, New York, USA, 2006), ACM Press.

[150] MORTENSEN, J., KHANNA, P., AND SLATER, M. Light field propagation and rendering on the GPU. In *Proceedings of the 5th international conference on Computer graphics, virtual reality, visualisation and interaction in Africa - AFRIGRAPH '07* (New York, New York, USA, 2007), ACM Press.

[151] MOSS, J. D., MUTH, E. R., TYRRELL, R. A., AND STEPHENS, B. R. Perceptual thresholds for display lag in a real visual environment are not

affected by field of view or psychophysical technique. *Displays 31*, 3 (jul 2010), 143–149.

[152] MYER, T. H., AND SUTHERLAND, I. E. On the design of display processors. *Communications of the ACM 11*, 6 (jun 1968), 410–414.

[153] NAVARRO, F., SERÕN, F. J., AND GUTIERREZ, D. Motion blur rendering: State of the art. *Computer Graphics Forum 30*, 1 (2011), 3–26.

[154] NEHAB, D., SANDER, P. V., LAWRENCE, J., TATARCHUK, N., AND ISIDORO, J. R. Accelerating Real-time Shading with Reverse Reprojection Caching. In *Proceedings of the 22Nd ACM SIGGRAPH/EUROGRAPHICS Symposium on Graphics Hardware* (2007), pp. 25–35.

[155] NG, A., LEPINSKI, J., WIGDOR, D., SANDERS, S., AND DIETZ, P. Designing for low-latency direct-touch input. In *Proceedings of the 25th annual ACM symposium on User interface software and technology - UIST '12* (New York, New York, USA, 2012), ACM Press, pp. 453–464.

[156] NVIDIA. NVIDIA® Tesla™ GPU Computing Technical Brief Version 1.0.0. Tech. rep., NVIDIA Corporation, 2007.

[157] NVIDIA. NVIDIA's Next Generation CUDA Compute Architecture: Fermi V1.1. Tech. rep., NVIDIA Corporation, 2009.

[158] NVIDIA. NVIDIA's Next Generation CUDA Compute Architecture: Kepler GK100. Tech. rep., NVIDIA Corporation, 2012.

[159] OCULUS VR, LLC. Rendering to the oculus rift. `https://developer3.oculus.com/documentation/pcsdk/0.5/concepts/dg-render/`, 2017.

[160] ODOM, C. N., SHETTY, N. J., AND REINERS, D. Ray Traced Virtual Reality. In *Advances in Visual Computing: Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2009, ch. Ray Traced, pp. 1031–1042.

[161] OLAFSDOTTIR, H. B., PERRAULT, S. T., AND GUIARD, Y. Fitts' with a Twist: An Exploration of Scale Effects using a New Experimental Paradigm. In *14ème congrès international de l'ACAPS* (2011).

[162] OUHYOUNG, M., AND WU, J.-R. On latency compensation and its effects on head-motion trajectories in virtual environments. *The Visual Computer 16*, 2 (mar 2000), 79–90.

[163] OWENS, J. D. *Computer Graphics on a Stream Architecture*. PhD thesis, Stanford University, 2002.

[164] PAN, Y., AND STEED, A. Preserving gaze direction in teleconferencing using a camera array and a spherical display. *3DTV-Conference* (2012), 1–4.

[165] PAPADAKIS, G., MANIA, K., AND KOUTROULIS, E. A system to measure, control and minimize end-to-end head tracking latency in immersive simulations. In *VRCAI '11 Proceedings of the 10th International Conference on Virtual Reality Continuum and Its Applications in Industry* (2011), pp. 581–584.

[166] PARK, K., AND KENYON, R. Effects of network characteristics on human performance in a collaborative virtual environment. *Proceedings of the 1999 IEEE Virtual Reality Conference* (1999).

[167] PARSONS, T. D., AND RIZZO, A. A. Affective outcomes of virtual reality exposure therapy for anxiety and specific phobias: A meta-analysis. *Journal of Behavior Therapy and Experimental Psychiatry 39*, 3 (2008), 250–261.

[168] PAVLOVYCH, A., AND STUERZLINGER, W. The tradeoff between spatial jitter and latency in pointing tasks. In *Proceedings of the 1st ACM SIGCHI symposium on Engineering interactive computing systems - EICS '09* (New York, USA, 2009), ACM Press, p. 187.

[169] PAVLOVYCH, A., AND STUERZLINGER, W. Target following performance in the presence of latency, jitter, and signal dropouts. In *Proceedings of Graphics Interface 2011 (GI '11)* (2011), pp. 33–40.

[170] PELL, O., AND AVERBUKH, V. Maximum Performance Computing with Dataflow Engines. *Computing in Science & Engineering 14*, 4 (jul 2012), 98–103.

[171] PELL, O., BOWER, J., DIMOND, R., MENCER, O., MEMBER, S., AND FLYNN, M. J. Finite-Difference Wave Propagation Modeling on Special-Purpose Dataflow Machines. *IEEE Transactions on Parallel and Distributed Systems 24*, 5 (2013), 906–915.

[172] PHILLIPS, L., INTERRANTE, V., KAEDING, M., RIES, B., AND ANDERSON, L. A Further Assessment of Factors Correlating with Presence in Immersive Virtual Environments. In *Proceedings of the 16th Eurographics conference on Virtual Environments & Second Joint Virtual Reality* (2010), pp. 55–63.

[173] PHILLIPS, L., INTERRANTE, V., KAEDING, M., RIES, B., AND ANDERSON, L. Correlations Between Physiological Response, Gait, Personality, and Presence in Immersive Virtual Environments. *Presence: Teleoperators and Virtual Environments 21*, 2 (2012), 119–141.

[174] PHILLIPS, L., RIES, B., INTERRANTE, V., KAEDING, M., AND ANDERSON, L. Distance perception in NPR immersive virtual environments, revisited. In *Proceedings of the 6th Symposium on Applied Perception in Graphics and Visualization - APGV '09* (New York, New York, USA, 2009), ACM Press, p. 11.

[175] PHILLIPS, L., RIES, B., KAEDI, M., AND INTERRANTE, V. Avatar self-embodiment enhances distance perception accuracy in non-photorealistic immersive virtual environments. In *Proceedings of the 2010 IEEE Virtual Reality Conference* (mar 2010), IEEE, pp. 115–1148.

[176] PHILLIPS, L., RIES, B., KAEDING, M., AND INTERRANTE, V. Gait Parameters in Stressful Virtual Environments. In *Proceedings of the 2nd IEEE VR 2010 Workshop on Perceptual Illusions in Virtual Environments* (2010), pp. 16–18.

[177] PLX TECHNOLOGY. PCI Express Packet Latency Matters. 2007.

[178] POPESCU, V., POPESCU, V., EYLES, J., EYLES, J., LASTRA, A., LASTRA, A., STEINHURST, J., STEINHURST, J., ENGLAND, N., ENGLAND, N., NYLAND, L., AND NYLAND, L. The warpengine: An architecture for the post-polygonal age. In *Proceedings of SIGGRAPH 2000* (2000), pp. 433–442.

[179] QU, H., QU, H., WAN, M., WAN, M., QIN, J., QIN, J., KAUFMAN, A., AND KAUFMAN, A. Image Based Rendering with Stable Frame Rates. In *Proceedings of the Conference on Visualization '00* (Salt Lake City, Utah, USA, 2000), pp. 251–258.

[180] REGAN, M., AND POSE, R. Priority rendering with a virtual reality address recalculation pipeline. In *Proceedings of the 21st annual conference on Computer Graphics and Interactive Techniques - SIGGRAPH '94* (New York, New York, USA, 1994), ACM Press, pp. 155–162.

[181] REGAN, M. J. P., MILLER, G. S. P., RUBIN, S. M., AND KOGELNIK, C. A real-time low-latency hardware light-field renderer. In *Proceedings of the 26th annual conference on Computer Graphics and Interactive Techniques - SIGGRAPH '99* (1999), pp. 287–290.

[182] REINERT, B., KOPF, J., RITSCHEL, T., CUERVO, E., CHU, D., AND SEIDEL, H.-P. Proxy-guided Image-based Rendering for Mobile Devices. *Computer Graphics Forum 35*, 7 (oct 2016), 353–362.

[183] REUSE, D. . Xilinx releases aurora open protocol to significantly reduce system cost and increase bandwidth. `http://www.design-reuse.com/news/4144/xilinx-aurora-open-protocol-significantly-system-cost-increase-bandwidth.html`, 2002.

[184] RIESER, J. J., HERBERT L. PICK, ASHMEAD, D. H., AND GARING, A. E. Calibration of Human Locomotion and Models of Perceptual-Motor Organization. *Journal of Experimental Psychology: Human Perception and Performance 21*, 3 (1995), 480–497.

[185] SAHM, C. S., CREEM-REGEHR, S. H., THOMPSON, W. B., AND WILLEMSEN, P. Throwing versus walking as indicators of distance perception in similar real and virtual environments. *ACM Transactions on Applied Perception 2*, 1 (2005), 35–45.

[186] SAMARAWEERA, G., PERDOMO, A., AND QUARLES, J. Applying latency to half of a self-avatar's body to change real walking patterns. In *Proceedings of 2015 IEEE Virtual Reality (VR)* (mar 2015), IEEE, pp. 89–96.

[187] SAMARAWEERA, G., RONGKAI GUO, AND QUARLES, J. Latency and avatars in Virtual Environments and the effects on gait for persons with mobility impairments. In *Proceedings of the 2013 IEEE Symposium on 3D User Interfaces (3DUI)* (mar 2013), IEEE, pp. 23–30.

[188] SANCHEZ-VIVES, M. V., AND SLATER, M. From presence to consciousness through virtual reality. *Nature reviews. Neuroscience 6*, 4 (apr 2005), 332–9.

[189] SARACINI, C., FRANKE, R., BLÜMEL, E., AND BELARDINELLI, M. O. Comparing distance perception in different virtual environments. *Cognitive Processing 10*, SUPPL. 2 (2009), 294–296.

[190] SCHERZER, D., YANG, L., MATTAUSCH, O., NEHAB, D., SANDER, P. V., WIMMER, M., AND EISEMANN, E. A Survey on Temporal Coherence Methods in Real-Time Rendering. *Eurographics 30*, 2 (2011), 1–29.

[191] SCHIRMACHER, H., VOGELGSANG, C., SEIDEL, H.-P., AND GREINER, G. Efficient Free Form Light Field Rendering. In *Proceedings of the Vision Modeling and Visualization Conference 2001 - VMV '01* (2001), pp. 249–256.

[192] SCHRODER, F. apE - The Original Dataflow Visualization Environment. *ACM SIGGRAPH Computer Graphics - Special focus: modular visualization environments (MVEs) 29*, 2 (1995), 5–9.

[193] SCRATCHAPIXEL. Ray-triangle intersection: Geometric solution. `http://www.scratchapixel.com/lessons/3d-basic-lessons/lesson-9-ray-triangle-intersection/ray-triangle-intersection-geometric-solution/`, 2011.

[194] SEOW, S. Information Theoretic Models of HCI: A Comparison of the Hick-Hyman Law and Fitts' Law. *Human-Computer Interaction 20*, 3 (sep 2005), 315–352.

[195] SEYMOUR, N. E., GALLAGHER, A. G., ROMAN, S. A., O'BRIEN, M. K., BANSAL, V. K., ANDERSEN, D. K., AND SATAVA, R. M. Virtual reality training improves operating room performance: results of a randomized, double-blinded study. *Annals of surgery 236*, 4 (2002), 458–63; discussion 463–4.

[196] SHEIKH, H. R., AND BOVIK, A. C. Image information and visual quality. *IEEE transactions on image processing: a publication of the IEEE Signal Processing Society 15*, 2 (2006), 430–444.

[197] SHERMAN, W. R., AND CRAIG, A. B. *Understanding Virtual Reality*. Morgan Kaufmann, 2002.

[198] SHERMAN, W. R., AND CRAIG, A. B. Understanding Virtual Reality. In *Understanding Virtual Reality*. Morgan Kaufmann Publishers, San Francisco, 2003.

[199] SHROUT, R. Oculus launches asynchronous spacewarp, 45 fps vr. `https://www.pcper.com/news/Graphics-Cards/Oculus-Launches-Asynchronous-Spacewarp-45-FPS-VR`, 2016.

[200] SHUM, H., AND KANG, S. A review of image-based rendering techniques. *IEEE/SPIE Visual Communications and Image Processing 2000 2* (2000), 1–12.

[201] SHUM, H.-Y., AND HE, L.-W. Rendering with concentric mosaics. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques - SIGGRAPH '99* (New York, New York, USA, 1999), ACM Press, pp. 299–306.

[202] SILICON LABORATORIES. AN562: PCI Express 3.0 Jitter Requirements. 1–24.

[203] SIMMONS, M. Ethernet Theory of Operation. 1–26.

[204] SLATER, M. A Note on Virtual Light Fields. Tech. rep., Univeristy College London, 2000.

[205] SLATER, M. Presence and The Sixth Sense. *Presence: Teleoperators and Virtual Environments 11*, 4 (2002), 435–439.

[206] SLATER, M. How Colorful Was Your Day? Why Questionnaires Cannot Assess Presence in Virtual Environments. *Presence Teleoperators and Virtual Environments 13*, 4 (2004), 484–493.

[207] SLATER, M. Place illusion and plausibility can lead to realistic behaviour in immersive virtual environments. *Philosophical Transactions of the Royal Society B: Biological Sciences 364* (dec 2009), 3549–3557.

[208] SLATER, M., ANTLEY, A., DAVISON, A., SWAPP, D., GUGER, C., BARKER, C., PISTRANG, N., AND SANCHEZ-VIVES, M. V. A virtual reprise of the Stanley Milgram obedience experiments. *PloS one 1*, 1 (jan 2006).

[209] SLATER, M., GUGER, C., EDLINGER, G., LEEB, R., PFURTSCHELLER, G., ANTLEY, A., GARAU, M., BROGNI, A., AND FRIEDMAN, D. Analysis of Physiological Responses to a Social Situation in an Immersive Virtual

Environment. *Presence: Teleoperators and Virtual Environments 15*, 5 (oct 2006), 553–569.

[210] SLATER, M., KHANNA, P., MORTENSEN, J., AND YU, I. Visual Realism Enhances Realistic Response in an Immersive Virtual Environment. *IEEE Computer Graphics and Applications 29*, 3 (2009), 76 – 84.

[211] SLATER, M., LOTTO, B., ARNOLD, M. M., AND SANCHEZ-VIVES, M. V. How we experience immersive virtual environments: the concept of presence and its measurement. *Anuario de Psicología 40*, 2 (2009), 193–210.

[212] SMIT, F., VAN LIERE, R., BECK, S., AND FROEHLICH, B. An Image-Warping Architecture for VR: Low Latency versus Image Quality. In *Proceedings of the 2009 IEEE Virtual Reality Conference - VR 2009* (mar 2009), Ieee, pp. 27–34.

[213] SMITH, G., SALZMAN, T., AND STUERZLINGER, W. 3D scene manipulation with 2D devices and constraints. In *Procdeedings GI '01* (2001), pp. 135–142.

[214] SO, R., AND CHUNG, G. Sensory Motor Responses in Virtual Environments: Studying the Effects of Image Latencies for Target-directed Hand Movement. In *2005 IEEE Engineering in Medicine and Biology 27th Annual Conference* (jan 2005), IEEE, pp. 5006–5008.

[215] SONG, D., AND GOLIN, E. Fine-grain visualization algorithms in dataflow environments. In *Proceedings Visualization '93* (1993), pp. 126–133.

[216] SOUAYED, R., AND GAITI, D. Experimental study of haptic interaction in distributed virtual environments. In *Proceedings of EuroHaptics 2004* (2004), pp. 260–266.

[217] STEINICKE, F., BRUDER, G., HINRICHS, K., LAPPE, M., RIES, B., AND INTERRANTE, V. Transitional environments enhance distance perception in immersive virtual reality systems. In *Proceedings of the 6th Symposium on*

Applied Perception in Graphics and Visualization - APGV '09 (2009), vol. 1, pp. 19–26.

[218] STEINICKE, F., BRUDER, G., HINRICHS, K., STEED, A., AND GERLACH, A. L. Does a gradual transition to the virtual world increase presence? In *Proceedings - IEEE Virtual Reality* (2009), pp. 203–210.

[219] SUFFERN, K. *Ray Tracing from the Ground Up.* A.K. Peters, Ltd., Wellesley, MA, 2007.

[220] SUTHERLAND, I. E. The Ultimate Display. In *Proceedings of the Congress of the Internation Federation of Information Processing (IFIP)* (1965), pp. 506–508.

[221] TEATHER, R. J., PAVLOVYCH, A., STUERZLINGER, W., AND MACKENZIE, I. S. Effects of tracking technology, latency, and spatial jitter on object movement. *2009 IEEE Symposium on 3D User Interfaces* (2009), 43–50.

[222] TEN HAGEN, P., HERMAN, I., AND DE VRIES, J. R. G. A dataflow graphics workstation. *Computers & Graphics 14*, 1 (1990), 83–93.

[223] TEOH, C., REGENBRECHT, H., AND O'HARE, D. The transmission of self: Body language availability and gender in videoconferencing. In *Proceedings of the 23rd Australian Computer-Human Interaction Conference, OzCHI 2011* (2011), pp. 273–280.

[224] TEXAS INSTRUMENTS. DS34RT5110-EVKH HDMI Extender Demo Kit for HDMI Cables User's Guide. Tech. Rep. SNLU033A, 2013.

[225] THOMPSON, W. B., CREEM-REGEHR, S. H., MOHLER, B. J., AND WILLEMSEN, P. Investigations on the interactions between vision and locomotion using a treadmill virtual environment. In *Proc. SPIE/IS&T Human Vision & Electronic Imaging Conference* (mar 2005).

[226] THOMPSON, W. B., WILLEMSEN, P., GOOCH, A. A., CREEM-REGEHR, S. H., LOOMIS, J. M., AND BEALL, A. C. Does the Quality of the Computer

Graphics Matter when Judging Distances in Visually Immersive Environments? *Presence: Teleoperators and Virtual Environments 13*, 5 (2004), 560–571.

[227] TODT, S., REZK-SALAMA, C., AND KOLB, A. Fast (Spherical) Light Field Rendering With Per-Pixel Depth. Tech. rep., University of Siegen, 2007.

[228] TRACING, G. P. U. R., GARANZHA, K., AND LOOP, C. Fast Ray Sorting and Breadth-First Packet Traversal for.

[229] UNITY. Unity documentation: Light probes. `http://docs.unity3d.com/Manual/LightProbes.html`, 2014.

[230] USOH, M., ARTHUR, K., WHITTON, M. C., BASTOS, R., STEED, A., SLATER, M., AND BROOKS, F. P. Walking > walking-in-place > flying, in virtual environments. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques - SIGGRAPH '99* (New York, New York, USA, 1999), ACM Press, pp. 359–364.

[231] USOH, M., CATENA, E., ARMAN, S., AND SLATER, M. Using Presence Questionnaires in Reality. *Presence: Teleoperators and Virtual Environments 9*, 5 (oct 2000), 497–503.

[232] VALVERDE, H. H. A review of flight simulator transfer of training studies. *Human Factors 15*, 6 (1973), 510–523.

[233] VESA. Vesa® adds 'adaptive-sync' to popular displayport™ video standard. `http://www.vesa.org/news/vesa-adds-adaptive-sync-to-popular-displayport-video-standard/`, 2014.

[234] VIDEO ELECTRONICS STANDARDS ASSOCIATION. VESA and Industry Standards and Guidelines for Computer Display Monitor Timing (DMT) Version 1.0 Revision 11, 2007.

[235] VINCENZI, D. A., DEATON, J. E., BUKER, T. J., BLICKENSDERFER, E. L., PRAY, R., AND WILLIAMS, B. Mitigation of System Latency in

Next Generation Helmet Mounted Display Systems. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* (sep 2011), vol. 55, pp. 2163–2167.

[236] VOITSECHOV, D., AND ETSION, Y. Single-graph multiple flows: Energy efficient design alternative for GPGPUs. In *Proceedings - International Symposium on Computer Architecture* (2014), pp. 205–216.

[237] WAJID, R., MANSOOR, A. B., AND PEDERSEN, M. A Human Perception Based Performance Evaluation of Image Quality Metrics. 303–312.

[238] WALD, I., IZE, T., KENSLER, A., KNOLL, A., AND PARKER, S. G. Ray tracing animated scenes using coherent grid traversal. *ACM Transactions on Graphics 25*, 3 (jul 2006), 485.

[239] WALD, I., MARK, W. R., GÜNTHER, J., BOULOS, S., IZE, T., HUNT, W., PARKER, S. G., AND SHIRLEY, P. State of the Art in Ray Tracing Animated Scenes. Tech. Rep. 6, EUROGRAPHICS 2007 State of the Art Reports, sep 2009.

[240] WARE, C., AND BALAKRISHNAN, R. Reaching for Objects in VR Displays: Lag and Frame Rate. *ACM Transactions on Computer-Human Interaction (TOCHI) 1*, 4 (1994), 331–356.

[241] WATSON, B., AND LUEBKE, D. Breaking the Frame: A New Approach to Temporal Sampling. Tech. rep., 1994.

[242] WATSON, B., SPAULDING, V., WALKER, N., AND RIBARSKY, W. Evaluation of the effects of frame time variation on VR task performance. In *Proceedings of the 1997 IEEE Virtual Reality Conference* (1997), IEEE Comput. Soc. Press, pp. 38–44.

[243] WHYTE, J. K. Industrial applications of virtual reality in architecture and construction. *Electronic Journal of Information Technology in Construction 8*,

Special Issue on Virtual Reality Technology in Architecture and Construction (2003), 43–50.

[244] WIDMER, S., PAJĄK, D., SCHULZ, A., PULLI, K., KAUTZ, J., GOESELE, M., AND LUEBKE, D. An adaptive acceleration structure for screen-space ray tracing. In *Proceedings of the 7th Conference on High-Performance Graphics - HPG '15* (New York, New York, USA, 2015), ACM Press, pp. 67–76.

[245] WILLEMSEN, P., COLTON, M. B., CREEM-REGEHR, S. H., AND THOMPSON, W. B. The effects of head-mounted display mechanical properties and field of view on distance judgments in virtual environments. *ACM Transactions on Applied Perception 6*, 2 (2009), 1–14.

[246] WILLIAMS, B., RASOR, T., AND NARASIMHAM, G. Distance perception in virtual environments. In *Proceedings of the 6th Symposium on Applied Perception in Graphics and Visualization - APGV '09* (New York, New York, USA, 2009), ACM Press, p. 7.

[247] WINTER, H. Memory Controller Pro. Tech. rep., Maxeler Technologies Ltd., 2010.

[248] WITMER, B. G., AND SADOWSKI, W. J. Nonvisually Guided Locomotion to a Previously Viewed Target in Real and Virtual Environments. *Human Factors: The Journal of the Human Factors and Ergonomics Society 40*, 3 (1998), 478–488.

[249] WITMER, B. G., AND SINGER, M. J. Measuring Presence in Virtual Environments: A Presence Questionnaire. *Presence 7*, 3 (1998), 225–240.

[250] WOLBERG, G. Image morphing: a survey. *The Visual Computer 14*, 8-9 (dec 1998), 360–372.

[251] WOODWORTH, R. S. Accuracy of voluntary movement. *The Psychological Review: Monograph Supplements* (1899).

[252] WOOP, S., SCHMITTLER, J., AND SLUSALLEK, P. RPU: a programmable ray processing unit for realtime ray tracing. *ACM Transactions on Graphics 1*, 212 (2005), 434–444.

[253] WU, J., YANG, J., AND HONDA, T. Human Movement Science Fitts' law holds for pointing movements under conditions of restricted visual feedback. *Human Movement Science 29*, 6 (2010), 882–892.

[254] WYNN, C. An Introduction to BRDF-Based Lighting. Tech. rep., NVIDIA Corporation, 2000.

[255] XILINX. Aurora 64B/66B Protocol Specification. Tech. rep., 2014.

[256] YANAGIDA, Y., INAMI, M., AND TACHI, S. Improvement of Temporal Quality of HMD for Rotational Motion. In *RO-MAN 1998 - The 16th IEEE International Symposium on Robot and Human Interactive Communication* (1998), pp. 121–126.

[257] YANG, L., TSE, Y.-C., SANDER, P. V., LAWRENCE, J., NEHAB, D., HOPPE, H., AND WILKINS, C. L. Image-based bidirectional scene reprojection. *ACM Transactions on Graphics 30*, 6 (dec 2011), 1.

[258] YONGSHENG, L., CHEN, S. L., WENHAO, Z., XIAOJUN, L., TAO, X., LIMIN, Z., SHIMING, Y., AND JINGFENG, C. The design methodology of a High-Performance dataflow supercomputer on a reconfigurable chipset for use in 3D graphics applications. In *Proceedings of the 2014 12th IEEE International Conference on Solid-State and Integrated Circuit Technology (ICSICT)* (2014).

[259] YOUNGBLUT, C. What a Decade of Experiments Reveals about Factors that Infl uence the Sense of Presence: Latest Findings. Tech. rep., Institute for Defence Analysis, 2007.

[260] ZAGIER, E. J. S. Defining and Refining Frameless Rendering. Tech. rep., University of North Carolina, Chapel Hill, 1996.

[261] ZHANG, X., AND WANDELL, B. A. A spatial extension of CIELAB for digital color image reproduction. *Journal of the Society for Information Display 5*, 1 (1997).

[262] ZHANG, X., AND WANDELL, B. A. Color image fidelity metrics evaluated using image distortion maps. *Signal Processing 70*, 3 (1998), 201–214.

[263] ZHENG, F., WHITTED, T., LASTRA, A., LINCOLN, P., STATE, A., MAIMONE, A., AND FUCHS, H. Minimizing Latency for Augmented Reality Displays: Frames Considered Harmful. *ISMAR 2014* (2014).

[264] ZIMMONS, P., AND PANTER, A. The Influence of Rendering Quality on Presence And Task Performance in a Virtual Environment. In *Proceedings of the 2003 IEEE Virtual Reality Conference* (2003), pp. 293–294.

[265] ZWICKER, M., GROSS, M., AND PFISTER, H. A Survey and Classification of Real Time Rendering Methods. Tech. rep., Mitsubishi Research Laboratories, 2000.